VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2017 Roman Drápela

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

AUTENTIZACE V IOT
AUTHENTICATION IN THE IOT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Roman Drápela**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Dzurenda**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Roman Drápela ***ID:*** 173639
Ročník: 3 ***Akademický rok:*** 2016/17

NÁZEV TÉMATU:

Autentizace v IoT

POKYNY PRO VYPRACOVÁNÍ:

V rámci bakalářské práce student analyzuje současný stav protokolů pro autentizaci v IoT a následně provede jejich srovnání. Na základě provedené analýzy implementuje vybraný protokol do autentizačního systému pro přístupové systémy, který bude využívat tzv. chytrá zařízení (smart devices), jako jsou mobilní telefony, hodinky a podobně. Pro autentizaci bude využito technologií NFC (Near Field Communication) a BLE (Bluetooth Low Energy).

DOPORUČENÁ LITERATURA:

[1] MURPHY, Mark L. Android 2: Průvodce programováním mobilních aplikací. Vyd. 1. Brno: Computer Press, 2011, 375 s. ISBN 978-80-251-3194-7.

[2] Menezes, A. J., van Oorschot, P. C., Vanstone, S. A.: Handbook of Applied Cryptography, CRC Press, 1997, ISBN 0-8493-8523-7

Termín zadání: 1.2.2017 ***Termín odevzdání:*** 8.6.2017

Vedoucí práce: Ing. Petr Dzurenda
Konzultant:

doc. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce se zabývá možnostmi autentizace v IoT. V úvodu práce jsou popsány vybrané autentizační protokoly z oblasti IoT. V další části jsou popsány technologie NFC a BLE, které nacházejí v oblasti IoT uplatnění. V rámci práce je prakticky implementován Multi-device protokol HDM16 na různých platformách (PC, mobilní telefon, SmartWatch). Protokol umožňuje autentizaci založenou na operacích nad eliptickými křivkami. Nad protokolem jsou provedeny dvě měření (simulované a praktické). V závěru jsou všechny výsledky zhodnoceny.

KLÍČOVÁ SLOVA

Autentizace, Autentizace v IoT, BLE, Bluetooth Low Energy, Bluetooth Smart, HCE, NFC, Near Field Communication, Autentizační protokoly, ECC, Eliptické křivky

ABSTRACT

Bachelor thesis focuses on authentication in the IoT. Some of the authentication protocols from the area of IoT are introduced at the beginning of the thesis. Next part of the thesis describes the NFC and BLE technologies, which can be applied in the IoT. Multi-device protocol is implemented using different devices (PC, mobile phone, SmartWatch). This protocol enables autentication based on the processes above the elliptic curves. Two sets of measurements (simulated and practical) are provided. In the final part of the thesis all the results are assessed.

KEYWORDS

Authentication, Authentication in the IoT, BLE, Bluetooth Low Energy, Bluetooth Smart, HCE, NFC, Near Field Communication, Authentication protocols, ECC, Eliptic curve

DRÁPELA, Roman *Autentizace v IoT*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2017. 66 s. Vedoucí práce byl Ing. Petr Dzurenda

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Autentizace v IoT“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Petru Dzurendovi za odborné vedení, motivaci, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

V této době se stále více rozvíjí myšlenka zvaná IoT (Internet věcí – Internet of Things). Teze představuje budoucí svět, ve kterém bude rozsah zařízení připojených k internetu bez hranic. Do sítě se budou moci připojit i zařízení, která pro nás do nedávné chvíle byla pouhým pasivním spotřebičem jako např. hodinky, pračka nebo popelnice. V době IoT bude např. lednička schopna komunikovat a odesílat oznámení o procházející expirační době výrobků v ní uložených nebo sníženém množství určených potravin. Komunikační entita nebude potřebovat velký výpočetní výkon pro běžné odeslání oznámení nebo vyhovění požadavku. Komunikační technologie, které budou zařízení využívat, nesmí být příliš náročné na spotřebu energie a přitom musí zachovat dostatečnou efektivitu. Zároveň s přibývajícím množstvím zařízení, které nás budou obklopovat, je potřebná jejich jednoznačná identifikace. Tento proces se nazývá autentizace.

Tato práce se bude věnovat autentizaci v oblasti Internetu věcí. Budou popsány také energeticky nenáročné technologie a protokoly, které mohou být pro tuto aplikaci použity. Konkrétně budou popsány technologie NFC (Near Field Communication) a BLE (Bluetooth Low Energy). V rámci práce bude také provedena analýza aktuálních autentizačních protokolů v IoT a jejich stručný popis. Nakonec bude vybraný protokol HDM16 i prakticky implementovaný. Na závěr bude provedeno měření kryptografických primitiv a praktické měření doby autentizace.

Výstupem práce bude zhodnocení energeticky nenáročných technologií a protokolů. Dále budou vytvořeny tři aplikace pro různá rozhraní (PC, mobilní telefon, SmartWatch), která budou komunikovat skrze již zmíněné technologie NFC a BLE a budou využívat protokol založený na kryptografii nad eliptickými křivkami. Nakonec bude uvedeno vyhodnocení výsledků měření s použitím konkrétních eliptických křivek a posouzení použití pro autentizaci v IoT.

1 KRYPTOGRAFIE

Kryptografie je věda, která studuje šifrovací algoritmy, kryptografické nástroje, hardwarové implementace šifrovacích algoritmů, kryptografické protokoly apod. Pomocí ní můžeme zajistit důvěrnost dat (šifrování), autentičnost dat, autentizaci dat a entit. Šifrování probíhá pomocí tzv. kryptografických klíčů, které se distribuují účastníkům komunikace. Umožňuje přenášet zprávy v nečitelné podobě. Pouze uživatel, který zná dešifrovací klíč, je schopen číst zprávy.

Při autentizaci je důležité, aby byl identifikátor unikátní pouze pro jeden objekt. S tím přichází riziko nepovolaného přístupu, kdy útočník použije identifikátor, který mu nepatří, proto zde má proces šifrování velmi důležitou roli.

Podle vztahu šifrovacího a dešifrovacího klíče můžeme obecně rozdělit kryptosystémy na symetrické nebo asymetrické. V kapitole Kryptografie bylo čerpáno z [1].

Symetrická kryptografie

Při použití symetrické kryptografie je šifrovací klíč buď stejný jako dešifrovací nebo je lehce odvoditelný. U tohoto důvodu musí obě strany držet klíč v tajnosti. Výhoda těchto systémů je jejich jednoduchost, délka kryptografických klíčů (kratší než v asymetrické kryptografii), rychlost a nízká výpočetní náročnost. Nevýhodou je bezpečná distribuce klíčů.

Asymetrická kryptografie

U těchto systémů jsou oba klíče různé, tzn. jeden klíč není odvoditelný z druhého. Je to založeno na problémech řešitelnosti některých matematických úloh. Z tohoto důvodu systémy poskytují tzv. nepopíratelnost. Uživatel, který vytvořil digitální podpis (pomocí soukromého klíče) to nemůže popřít, jelikož pouze on zná soukromí klíč, nikdo jiný. Při distribuci klíčů stále hrozí útok MITM (Man-in-the-middle – útok na kryptografii). Pro distribuci veřejného klíče je potřeba využít infrastrukturu veřejných klíčů PKI (Public Key Infrastructure). Nevýhodou je vzhledem ke složitosti delší čas šifrování a dešifrování zpráv.

1.1 Autentizace

Autentizace je pojem, se kterým se potkáváme ve spojitosti s informační bezpečností běžně. Jde o proces ověření totožnosti určitého objektu. V dřívější době byl proces automatizace spojovaný hlavně s jednoznačnou identifikací osob (osoba → zařízení

určené pro autentizaci). S rostoucím počtem zařízení, které mohou komunikovat, je však potřeba zjistit i jejich autentičnost (původnost). Ověřit lze entity (zařízení, osoby, procesy) ale i data. Autentizace tedy probíhá vzájemně (identifikované zařízení → identifikující zařízení).

Všeobecně lze říci, že autentizovat se dá třemi způsoby:

1. Autentizace znalostí – identifikace je na základě určité znalosti (PIN, heslo atd.).
2. Autentizace předmětem – identifikace je na základě určitého předmětu (čipová karta, identifikační průkaz atd.).
3. Autentizace biometrikou – identifikace je na základě určitého charakteru žadatele o autentizaci (otisk prstu, oční sítnice atd.).

Způsoby autentizace se dají pro vyšší zabezpečení i kombinovat, potom mluvíme o tzv. multifaktorové autentizaci. Ukázkově například v internet banking, kde je spojená autentizace znalostí (přístupové heslo) a předmětem (SMS zpráva).

1.2 Šifrování

Šifrovací systémy dělíme do dvou tříd, podle vztahu šifrovacího a dešifrovacího klíče. Podle toho je pak šifrování symetrické nebo asymetrické.

Symetrická kryptografie

Při použití symetrického šifrování je šifrovací klíč buď stejný jako dešifrovací nebo je lehce odvoditelný. U tohoto důvodu musí obě strany držet klíč v tajnosti. Výhoda těchto systémů je jejich jednoduchost, délka kryptografických klíčů (kratší než v asymetrickém šifrování), rychlost a nízká výpočetní náročnost. Nevýhodou je bezpečná distribuce klíčů.

Asymetrická kryptografie

U těchto systémů jsou oba klíče různé, tzn. jeden klíč není odvoditelný z druhého. Je to založeno na problémech řešitelnosti některých matematických úloh. Z tohoto důvodu systémy poskytují tzv. nepopíratelnost. Uživatel, který vytvořil digitální podpis (pomocí soukromého klíče) to nemůže popřít, jelikož pouze on zná soukromí klíč, nikdo jiný. Při distribuci klíčů stále hrozí útok MITM (Man-in-the-middle – útok na kryptografii). Pro distribuci veřejného klíče je potřeba využít infrastrukturu veřejných klíčů PKI (Public Key Infrastructure). Nevýhodou je vzhledem ke složitosti delší čas šifrování a dešifrování zpráv.

2 AUTENTIZAČNÍ PROTOKOLY V IOT

„Chytrá“ zařízení, jako například smart telefony a různé snímací uzly, dnes tvoří rozvíjející se globální a Internet-based informační služby zvané IoT (Internet věcí – Internet of Things). Pro ověření autentičnosti používáme autentizační protokoly, což je několikastranný algoritmus definovaných kroků, které musí dvě a více stran podstoupit pro úspěšnou identifikaci. Protože v IoT počítáme s implementací zařízení s nízkým hardwarovým vybavením, je potřeba předpokládat z jedné strany co nejnižší potřebný úkon, aby jej bylo zařízení schopné v rozumném čase provést a zároveň neztratila autentizace na bezpečnosti. V této sekci probereme pár konkrétních protokolů, které jsou vhodné pro implementaci do IoT.

2.0.1 Protokol MQTT

Základní protokol MQTT (Message Queuing Telemetry Transport) již v roce 1999 vytvořil Dr. Andy Stanford-Clark z firmy IBM. První vydaná publikace o tomto protokolu vyšla až jako jeho třetí lehce poupravená verze [2]. Jedná se o textově založený protokol, který vyniká svojí jednoduchostí, maximálně odlehčenou potřebou zpráv, možností použití v prostředí s vysokou latencí a nespolehlivou sítí. Tento centralizovaný protokol byl původně vyvíjen pro použití v sítích TCP/IP (Transmission Control Protocol/Internet Protocol) ekosystému M2M (Machine to machine), nicméně díky svojí verzi MQTT-SN (MQTT for Sensor Networks) lze efektivně využít i v IoT pro zařízení bez TCP/IP.

Protokol je založen na tzv. zveřejňování/odebírání (publish/subscribe). Důležitý je zde centrální bod (MQTT broker), který se stará o výměnu dat. Zprávy jsou rozdělovány hierarchicky do témat (topic). Jelikož je protokol čistě textový, je důležité použití SSH/TLS (Secure Sockets Layer/Transport Layer Security), jinak by byla komunikace nešifrovaná [3, 4].

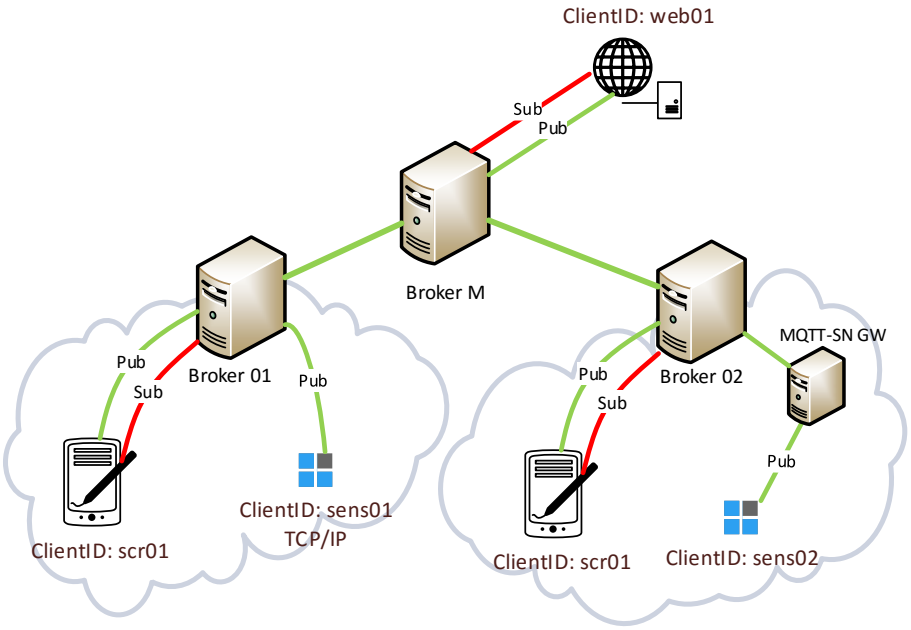
Infrastruktura

Jak již bylo řečeno, o distribuci dat se stará tzv. Broker. Jedná se o softwarovou aplikaci, která řídí mj. autentizaci a autorizaci klientů a mezi jeho funkce spadá i komunikace s dalšími Brokery. U zařízení nepodporujících TCP/IP musíme na straně Brokera použít modul MQTT-SN Gateway, který zprostředkovává komunikaci mezi oběma rozhraními.

Obrázek 2.1 zobrazuje infrastrukturu, kde je možné využít protokol MQTT. Na obrázku je ve středu hlavní Broker (Broker M), na kterého jsou připojeny tři cesty. Jednou cestou je připojen externí klient přes webové rozhraní (ClientID: web01),

ostatní dvě jsou LAN sítě s vlastním Brokerem (Broker 01 a 02), který řídí proud dat od a k hlavnímu Brokeru a zároveň zpracovává data ze svojí LAN sítě. Zařízení připojená na Broker můžou zprávy vysílat (Pub – publish) nebo přijímat (Sub – subscribe). U zařízení typu senzor předpokládáme jenom vysílání, protože by neuměl přijatá data zpracovat.

Zařízení požadující data (např. ClientID: scr01) nejdříve sestaví spojení se senzorem (např. ClientID: sens01 TCP/IP), příp. jiným vysílajícím klientem. Při sestavování musí prokázat identitu a pokud je důkaz identity, akceptován může komunikace probíhat dál. Klient následně vybere témata, která chce „odebírat“ a očekává odpovědi.



Obr. 2.1: Infrastruktura protokolu MQTT

MQTT je velmi snadno použitelný – existuje mnoho implementací brokerů (asi nejznámější a nejpoužívanější je open-source MQTT broker Mosquitto) a ještě víc klientských knihoven pro nejrůznější jazyky (od Javy přes Python, JavaScript, Ruby, Go až po Erlang) a zařízení (Arduino, mbed, netduino atd.). S protokolem MQTT se lze setkat i u cloudových služeb (AWS IoT, Azure IoT) nebo naopak u různých systémů pro domácí automatizaci (Domoticz) či nástrojů pro smartfony [4].

2.0.2 Protokol PAuthKey

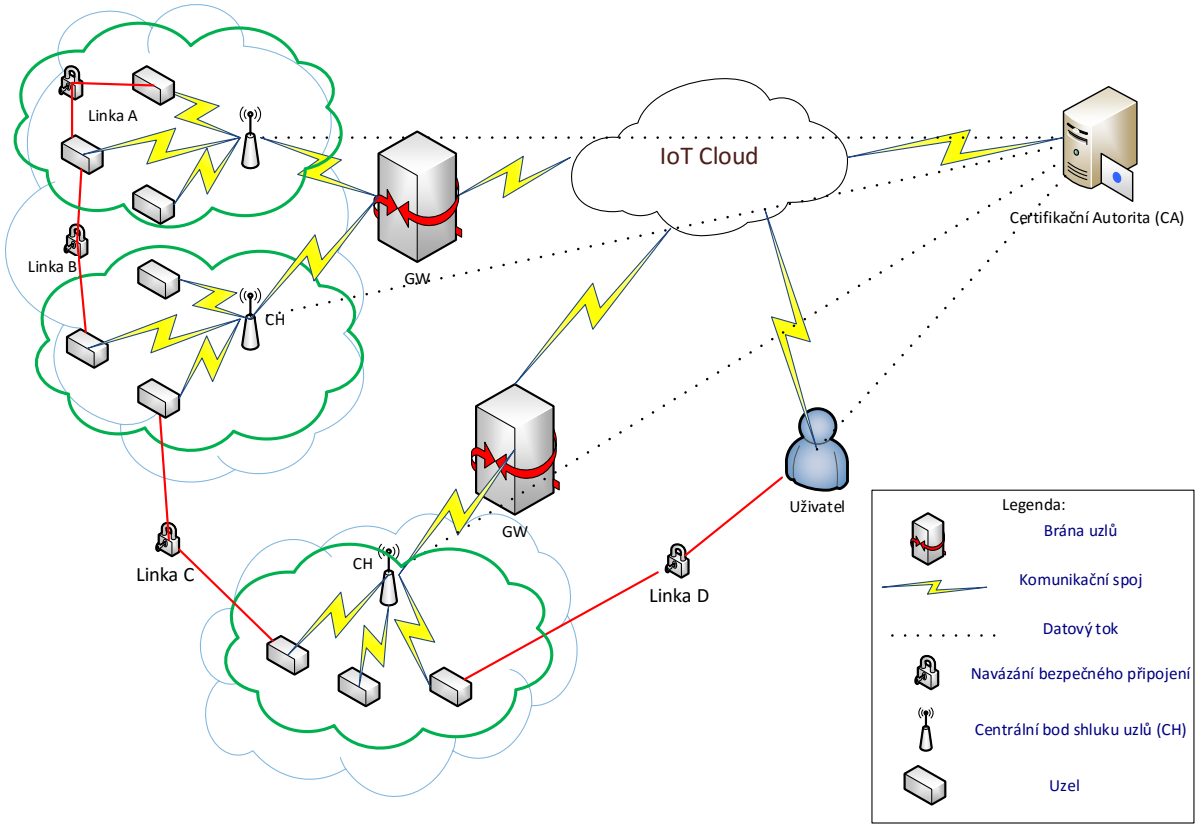
PAuthKey (Pervasive Authentication Protocol and Key Establishment) je relativně nový autentizační protokol a systém pro ustanovení klíčů určený především pro WSN

(Wireless Sensor Networks – Bezdrátová síť senzorů) v aplikacích IoT. Protokol byl představený v časopise International journal of distributed sensor networks v roce 2014 [5]. Protokol probíhá ve dvou fázích – fáze registrace a fáze autentizace. Je založen na kryptografii eliptických křivek. Prakticky se používá pro sítě typu WSN v distribuovaných IoT aplikacích.

Komunikace může být ve čtyřech možných variantách:

1. Dva senzory jsou umístěny v rámci stejného „shluku“ (cluster) (Komunikační linka A).
2. Dva senzory jsou umístěny ve stejné WSN ale v jiném „shluku“ (Komunikační linka B).
3. Dva senzory jsou umístěny v různé WSN i různém „shluku“ (Komunikační linka C).
4. Koncový uživatel je připojen na uzel (Komunikační linka D).

Obrázek 2.2 demonstruje jak může vypadat infrastruktura WSN a všechny možné druhy komunikace.



Obr. 2.2: Infrastruktura WSN v rámci demostrace protokolu PAuthKey

Před začátkem samotné autentizace dvou síťových entit je nutné podstoupit proces

registrace každé strany komunikace, a tím obdržení bezpečnostních kryptografických pověření. Později se získané pověření využije pro vzájemné ověření identity. Získat ho musí od důvěryhodné třetí strany, jako například CA (Certifikační Autorita). Předpokládá se, že CA je server velmi „bohatý“ na prostředky a je již známý pro okrajové uzly během fáze registrace.

V této architektuře můžeme najít dva typy síťových subjektů, jako jsou subjekty bohaté na prostředky (tj. koncoví uživatelé a centrální body shluků) a subjekty velmi omezené na prostředky (tj. okrajové uzly – senzory). Tato stromová topologie obsahuje centrální body shluků (cluster head – CH) jako kontrolní prvek pro každý uzel. Proto se má za to, že CH se v rámci jednoho shluku pro jednotlivé uzly zachovává jako CA - vydává implicitní certifikáty. Jak je vidět na obrázku 2.2, entity bohaté na prostředky nejdříve komunikují se společným CA již navázaným spojením skrze IoT cloud. Jestliže chce uživatel komunikovat s krajovým uzlem, musí nejdříve navázat zabezpečené DTLS (Datagram Transport Layer Security) s příslušným centrálním bodem shluku a získat od něj implicitní certifikáty. Poté použije získané certifikáty pro komunikaci s krajovým uzlem. Držení platného certifikátu umožňuje dvěma subjektům vzájemnou autentizaci bez ohledu na jejich lokální síť. Při změně umístění uzlů se dynamicky zažádá o nový certifikát. Díky tomu lze jednoduše přidávat a odebírat uzly v síti.

2.0.3 Protokol CoAP a DTLS

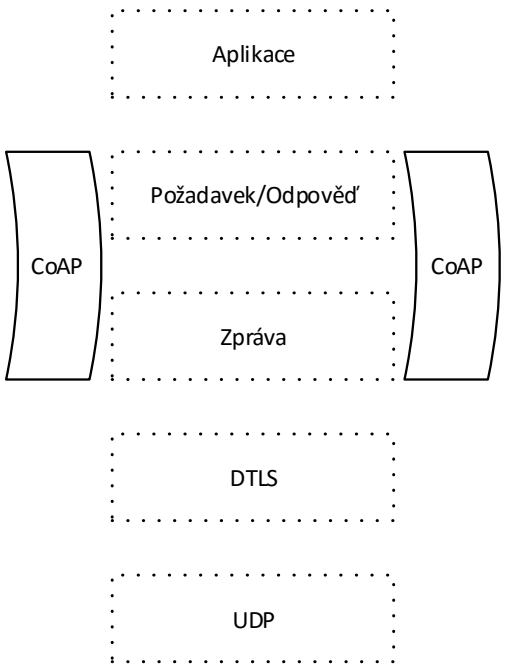
Protokol CoAP

Protokol CoAP (The Constrained Application Protocol) je specializovaný webově přenášený protokol uplatňující se u zařízení s omezenými prostředky jako je malá výpočetní rychlost (typicky 8-bitový mikrokontrolér s malou pamětí ROM i RAM) v sítích s vysokými ztrátami a propustností v desítkách kb/s. Je vyvinut pro aplikace typu M2M. CoAP je založený na modelu požadavek/odpověď. Díky obsaženým konceptům Webu, jako například URI (Uniform Resource Identifier) a různé typy internetových médií, snadno pracuje v rozhraní HTTP (Hypertext Transfer Protocol). Podporuje multicast, požaduje minimální režii a je jednoduchý pro zařízení s omezenými prostředky.

Model CoAP protokolu by se dal přirovnat k modelu klient/server u HTTP, zde se však zařízení chovají jako oboje. CoAP požadavek (request) je shodný s HTTP (HTTP request) a je odesílán klientem, jako požadavek na nějakou činnost, na server (identifikovaný URI). Server odešle odpověď s kódem požadavku. Na rozdíl od HTTP však CoAP přenáší data asynchronně přes datagramově-orientovaný přenos jako je UDP (User Datagram Protocol) [6].

CoAP definuje čtyři typy zpráv:

1. Confirmable („Potvrzení-schopný“) – na tento typ zprávy je požadovaný ACK (příp. Reset).
2. Non-confirmable („Potvrzení-neschopný“) – na tento typ zprávy není požadovaný ACK. Typicky opakované posílání požadavku (např. čtení hodnoty senzoru).
3. Acknowledgement (ACK, „Potvrzení přijetí“) – tento typ zprávy specifikuje pouze to, že zpráva typu Confirmable byla přijata. Neindikuje tedy úspěch ani neúspěch požadavku.
4. Reset – indikuje přijetí zprávy (Confirmable nebo Non-confirmable), ale pro správné zpracování požadavku chybí určité souvislosti. Také lze jednoduše odesláním prázdné zprávy Confirmable zjistit životnost uzlu tzv. „CoAP ping“.



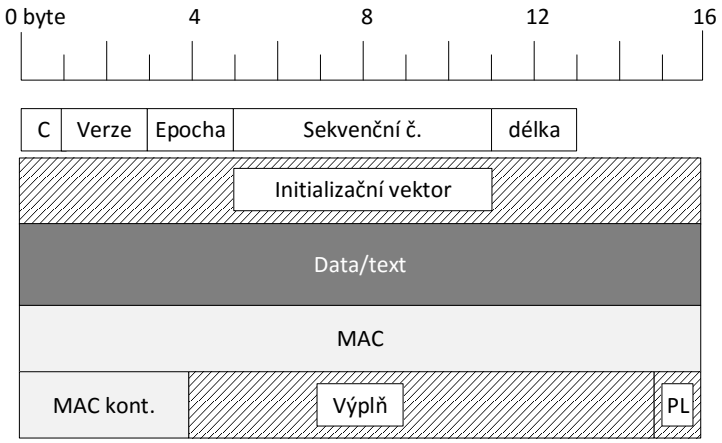
Obr. 2.3: Abstraktní vrstvy CoAP Protokolu

CoAP by se dal logicky považovat za použití přístupu založeného na dvouvrstvém modelu (obr. 2.3). Vrstva CoAP zpráv je přenášena přes protokol UDP s asynchronní povahou interakcí. Stejně jako HTTP je zabezpečeno TLS přes TCP Protokol, je CoAP zabezpečen DTLS přes UDP protokol. Konfigurace jsou však minimální, aby mohlo být DTLS použito i v omezených podmínkách.

Protože mohou různé požadavky vysílat pouze určité autority, je potřeba určité ověření identity. To je prováděno pomocí **protokolu DTLS**.

Protokol DTLS

Všechny zprávy přenášené přes protokol DTLS (Datagram Transport Layer Security) mají vložené 13 bytové DTLS záhlaví. Záhlaví specifikuje obsah zprávy (data aplikace, handshake data atd.), verzi protokolu, 64-bitovou sekvenci a délku záhlaví (Na obrázku 2.4 bílá část) [7].



Obr. 2.4: DTLS s CBC blokovou šifrou

Záhlaví je buď následováno prostým textem (daty), pokud zatím nebyla sjedaná žádná ochrana, nebo DTLS blokovou šifrou. Jestliže je bloková šifra použita, za záhlaví DTLS se umístí náhodný initializační vektor (IV), který má velikost jako bloková šifra. Chrání tak před útoky, kdy útočník vybírá adaptivně prostý text. Text je následovaný Hash zprávou HMAC (Hash-based Message Authentication Code), která přijímači dovoluje detekovat změnu DTLS záznamu. Zpráva je na konci výplní uměle prodloužena na délku blokové šifry. Šedá oblast na obrázku 2.4 je použitá pro šifru, vyšrafované části k výpočtu HMAC použity nejsou.

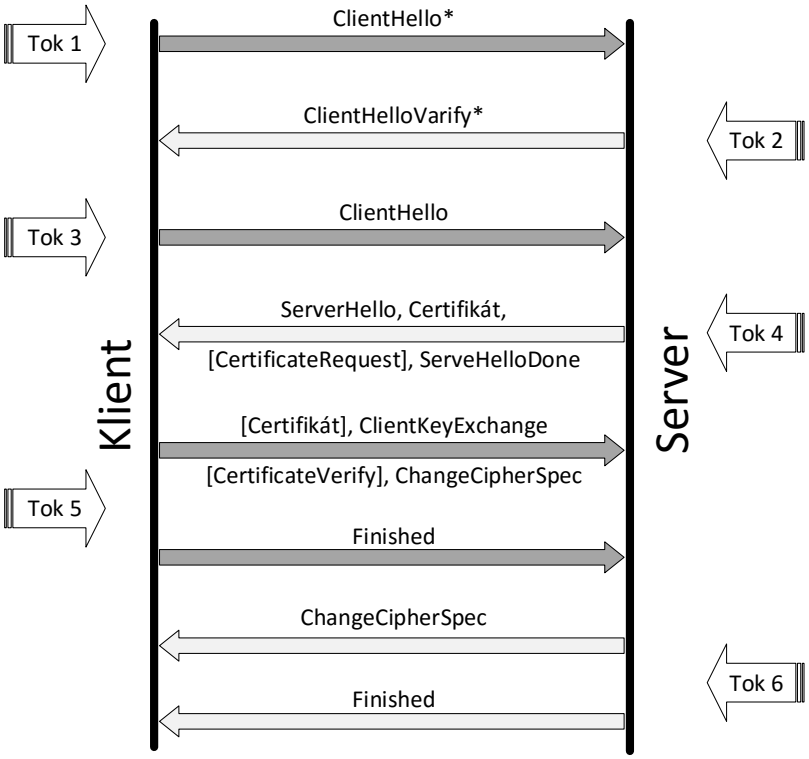
DTLS vychází z protokolu TLS, ale je přenášen místo přes TCP přes UDP a pro tento režim je i upraven. Nemusí tedy sestavovat spojení, předpokládá se ztráta paketů, jejich duplikování nebo doručení ve špatném pořadí (každý paket obsahuje pořadové číslo). Princip fáze Handshake však zůstává stejný jako u protokolu TLS [8]. Na rozdíl od TLS, DTLS nepodporuje stream šifer, protože jsou citlivé na jakékoliv ztráty. Namísto toho používá mód zřetězení šifrovaných bloků CBC (Cipher-Block Chaining). Klíčový materiál a šifrovací „sada“ (bloková šifra a HMAC) jsou sjednány mezi klientem a serverem během tzv. Handshake DTLS fáze.

Fáze Handshake má tři druhy ověření:

1. Neautentizovaný – žádná strana neprokazuje identitu.
2. Serverově autentizovaný – identitu prokazuje strana serveru klientovi.

3. Plně autentizovaný – identitu prokazuje strana klienta serveru.

Existují různé algoritmy, které mohou být použity pro ověření ve fázi Handshake, např. ECC nebo RSA. V další části uvažujeme ověření založené na RSA používané v systémech PKC (Public-Key Cryptography – Kryptografie s věřejným klíčem) jako je tomu například i u protokolu CoAP (viz Protokol CoAP). Certifikáty zajišťuje důvěryhodná třetí strana, jako je například komerční certifikační autorita (CA).



Obr. 2.5: DTLS handshake

Obrázek 2.5 zobrazuje plnou autentizaci. Jednotlivé zprávy jsou seskupovány do „toků“ v závislosti na jejich směru a výskytu sekvencí. Tok 1 a 2 jsou volitelné funkce na ochranu serveru proti DoS (Denial of Service) útokům. Klient musí prokázat, že může příjmat i odesílat data opětovným posláním zprávy `ClientHello`. Ta obsahuje šifrovací sady a verzi protokolu, které podporuje. Server odpovídá zprávou `ServerHello`, která obsahuje šifrovací sadu vybranou ze seznamu nabízené ze strany klienta. Také odesílá certifikát (X.509) aby ověřil svojí identitu následovaný zprávou požadující certifikát `CertificateRequest`. Zprávou `ServerHelloDone` indikuje server konec toku 4. Je-li na straně klienta přijat požadavek na certifikát a klient tuto službu podporuje, odešle jej na začátku toku 5. `ClientKeyExchange` zpráva obsahuje „hlavní tajemství“ (pre-master secret). Z tohoto tajemství se pak počítají jednot-

livé klíče a inicializační vektory, tj. klíče uživatele (šifrovací, dešifrovací, inicializační vektor) a klíče serveru (šifrovací, dešifrovací, inicializační vektor). Klíčovací materiál je následně z tajné zprávy odvozen. Vzhledem k tomu, že je polovina tajné zprávy zašifrovaná pomocí veřejného klíče, může fázi Handshake dokončit pouze v případě, pokud je ve vlastnictví soukromého klíče odpovídající veřejnému klíči serveru. Ve zprávě `CertificateVerify` klient autentizuje sebe tím, že je v držení soukromého klíče odpovídající veřejnému klíči klienta. `ChangeCipherSpec` znamená, že všechny následující zprávy směřující od klienta budou šifrovány pomocí sjednané šifrovací sady a klíčového materiálu. Zpráva `Finished` (ze strany klienta) obsahuje zašifrovaný přehled všech předchozích zpráv, čímž zajistí, že operace obou stran budou založené na stejných principech. Server odpoví s vlastní zprávou `ChangeCipherSpec` a zakončí handshake zprávou `Finished`.

2.0.4 Protokol HDM16

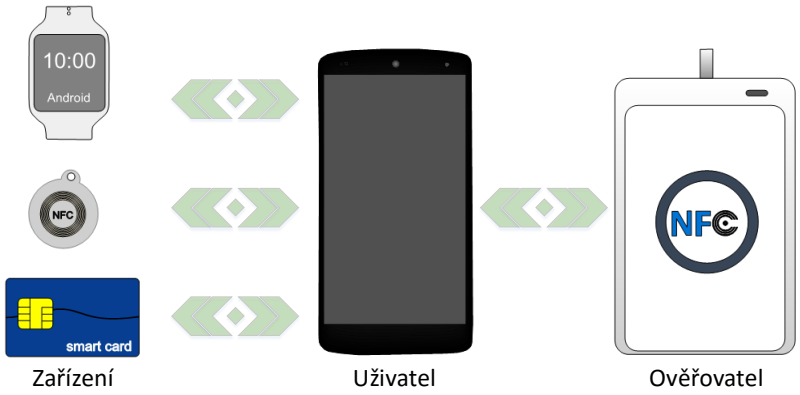
Protokol HDM16 (Multidevice protokol – Hajný, Dzurenda, Malina, 2016) byl představen týmem vývojářů z VUT v Brně v roce 2016 v článku [9]. Spočívá v autentizaci na základě zjištěné přítomnosti jiných zařízení. Využívá rozvíjejícího se konceptu IoT, kdy se předpokládá, že u sebe bude mít uživatel více jak jedno zařízení schopné komunikace a určitých výpočetních schopností. Původní verze protokolu je založena na problému diskrétního logaritmu a výpočtech v konečných tělesech. Bakalářská práce implementuje tento protokol, avšak problém je překlopen do oblasti eliptických křivek.

Protokol rozlišuje tři role:

- **Ověřovatel (Verifier)** – většinou se jedná o službu, která slouží k ověření identity uživatele (např. čtečka čipových karet u vstupu do objektu).
- **Uživatel (Master Device)** – zákazník je reprezentován jedním vybraným master-zařízením, které slouží v dalších krocích mj. jako jistá brána mezi Ověřovatelem a Zařízeními (např. mobilní telefon vybavený technologií NFC pro komunikaci se čtečkou).
- **Zařízení (Device)** – vázaná osobní zařízení (chytré hodinky, senzory, ...), která jsou zapojena do procesu autentizace k posílení bezpečnosti. Tato zařízení komunikují s hlavním zařízením (Master device), které provádí autentizaci s Ověřovatelem. Komunikace je prováděna např. přes technologii BLE.

Tyto role se angažují v následujících protokolech:

- **Nastavení (Setup Protocol)** – výměna inicializovaných parametrů mezi Ověřovatelem a Uživatelem. Předchází procesu autentizace.
- **Ověření (Authenticate Protocol)** – samotný proces autentizace. Probíhá mezi všemi entitami, které jsou zapojeny. Uživatel prokazuje znalost soukromých klíčů (každého zařízení). Ověřovatel buď důkaz identity akceptuje nebo zamítne.
- **Registrace (Register Protocol)** – uživatel přihlašuje do systému další zařízení (zajišťuje pro ně vlastní privátní klíč). Zapojeny musí být opět všechna registrovaná zařízení.
- **Odhlášení (Deregister Protocol)** – odhlášení zařízení ze systému, např. z důvodu ztráty, odcizení nebo poruchy. Do procesu musí být zapojena všechna zbývající registrovaná zařízení.



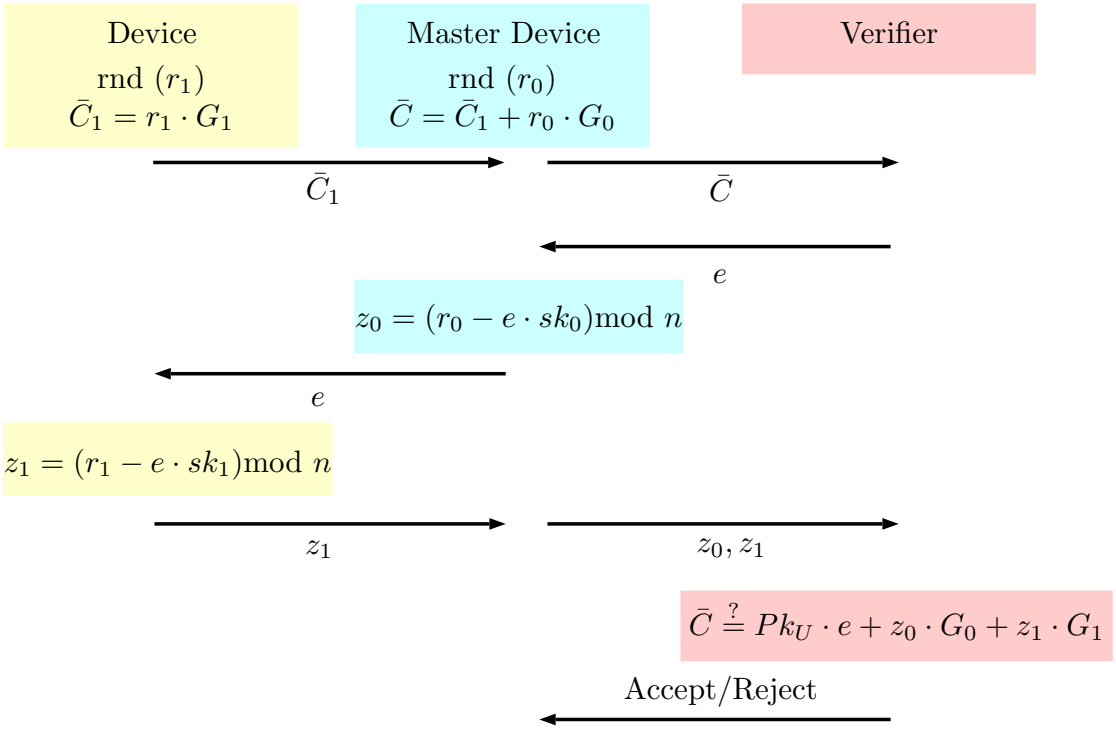
Obr. 2.6: Ukázka hierarchie komunikace v protokolu HDM16

Výhoda protokolu je, že uživatel má více zařízené a při procesu autentizace musí předložit všechna tato zařízení. Při ztrátě jednoho zařízení se nic neděje, jelikož útočník nemá telefon a všechna ostatní periferní zařízení.

Protokol HDM16 zajišťuje:

- **Spolehlivost (soundness)**, tj. pokud je uživatel skutečný, jeho identita bude prokázána.
- **Úplnost (completeness)**, tj. pokud je uživatel nečestný, dojde k jeho zamítnutí.
- **Nulovou znalost (zeroknowledge)**, tj. během procesu autentizace nejsou uvolněny žádné informace o soukromém klíči Uživatele (Master Device), pouze informace, zda uživatel zná klíč či nikoli.

Další výhodou je oproštění zákazníka od nutné identifikace znalostí nějakého klíče (PINu, hesla atd.) a autentizace může probíhat bez jeho zásahu. Protokol HDM16 je tedy bezpečný, jednoduše implementovatelný a zároveň pohodlný pro uživatele.



Obr. 2.7: Autentizační část protokolu HDM16 využívající eliptické křivky

Na obrázku 2.7 je zobrazena komunikace v autentizační části protokolu HDM16 využívající eliptické křivky s jedním zaregistrovaným zařízením (Device). Autentizaci probíhá mezi entitami Device (Zařízení), Master Device (Uživatel) a Verifier (Ověřovatel). Proces Nastavení předchází procesu Autentizace, ve kterém je specifikovaná konkrétní eliptická křivka. Tímto protokolem se nebudeme v rámci této práce zabývat a předpokládáme jeho úspěšné ukončení.

Velká písmena reprezentují body dané eliptické křivky, malá písmena jsou velká čísla (proměnné BigInteger), kde r_n reprezentují náhodně vygenerovaná čísla bitové délky eliptické křivky. Proměnná n je řád bodu eliptické křivky

V prvním kroku je v Zařízení (Device) vygenerováno náhodné číslo r_1 a vypočítán bod

$$\bar{C}_1 = r_1 \cdot G_1, \tag{2.1}$$

kde G_1 je bod eliptické křivky. Výsledný bod $\bar{C}_1$ je poslán do hlavního zařízení, tj. Uživatele (Master Device). Ten k tomuto výsledku přičte výsledek téže operace, ale s rozdílným bodem (G_0) a náhodným číslem (r_0) vygenerovaném na tomto zařízení. Ověřovatel po obdržení této zprávy pošle na Uživatele hash e jako výzvu.

Uživatel vypočítá ověřovací zprávu operací

$$z_0 = (r_0 - e \cdot sk_0) \text{mod } n, \tag{2.2}$$

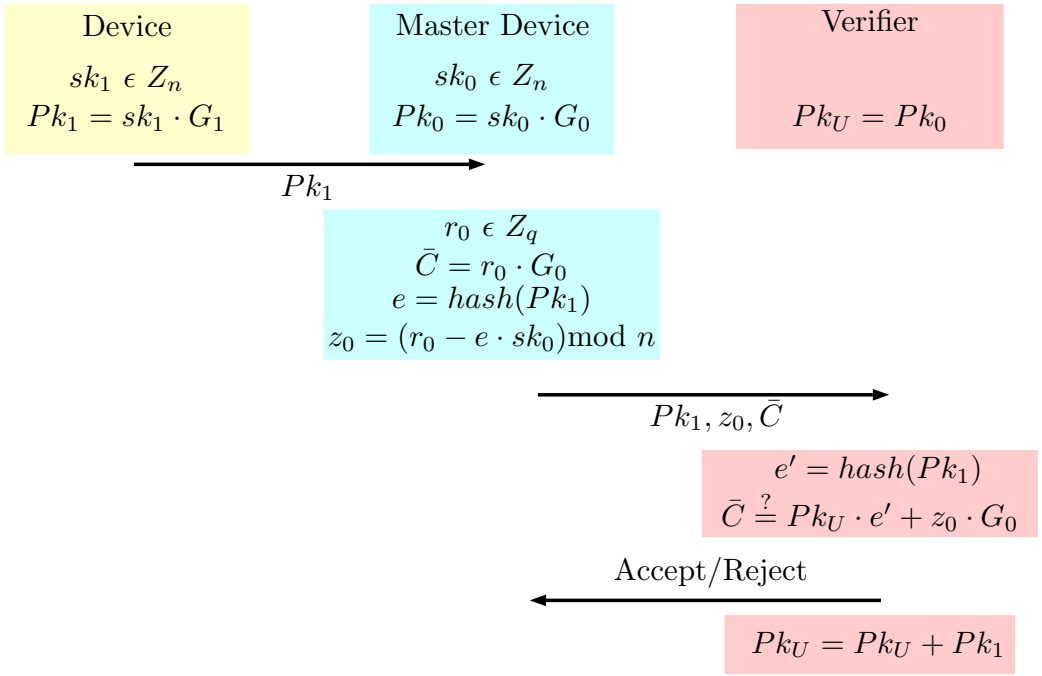
kde sk_0 je privátní klíč Uživatele a mod n je modulární operace s číslem hodnoty řádu bodu použité eliptické křivky. Hash e je přeposlán na Zařízení a ten provede shodnou operaci jako Uživatel, ale s vlastním privátním klíčem a vygenerovaným číslem r_1. Zpráva je poslána zpět do Uživatele a odtud obě zprávy putují za Ověřovatelem. Ten operací

$$\bar{C} \stackrel{?}{=} Pk_U \cdot e + z_0 \cdot G_0 + z_1 \cdot G_1 \tag{2.3}$$

ověří důkazy identit, kde

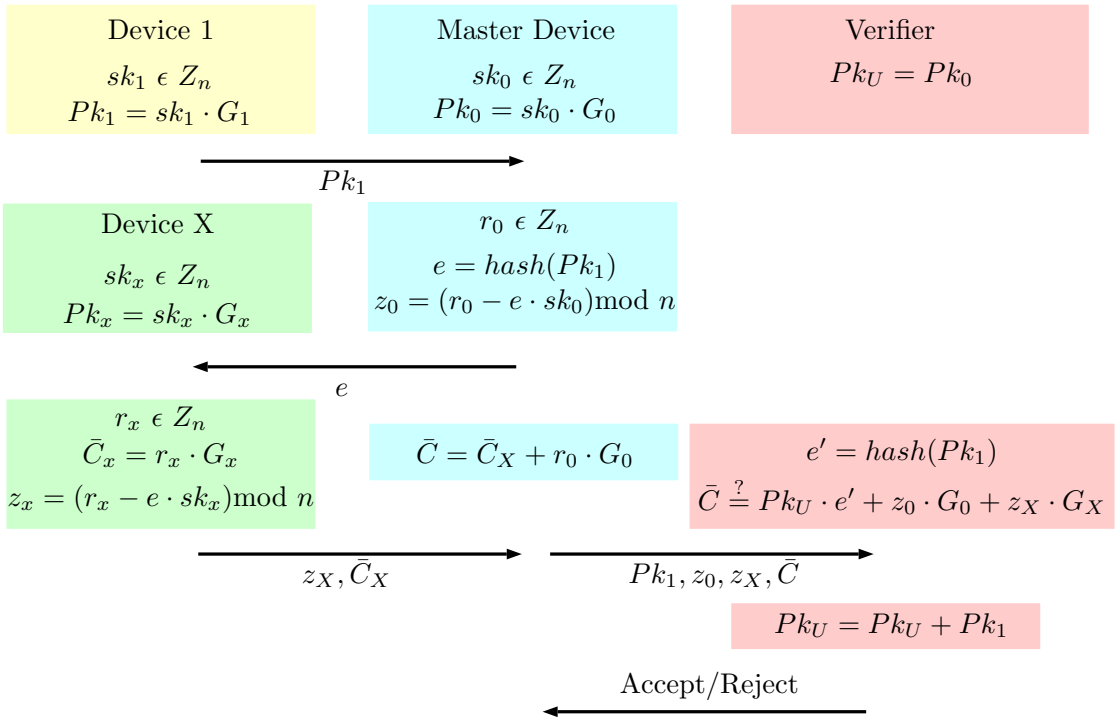
$$Pk_U = sk_1 \cdot G_1 + sk_0 \cdot G_0. \tag{2.4}$$

Body eliptické křivky $G_0 \ \ldots \ G_x$ jsou distribuovány Nastavujícím protokolem. V posledním kroku PC zhodnotí výsledek operací a vyhodnotí autentizaci.



Obr. 2.8: Registrační část protokolu HDM16 – příklad registrace prvního zařízení.

Další částí protokolu HDM16 je registrační část. Jedná se o modifikovanou část Autentizace, kde je pro zaregistrování nového zařízení potřeba získat jeho veřejný klíč (Pk_1) a provést ověření identity pomocí již zaregistrovaných zařízení. Registrace prvního zařízení (viz obr. 2.8) je nejrychlejší, protože se k ověření identity použije pouze hlavní zařízení Master device (Uživatel) a žádné jiné zařízení zaregistrované není.



Obr. 2.9: Registrační část – příklad registrace více zařízení.

Registrace nového zařízení, kdy jsou zaregistrované i jiné (viz obr. 2.9) proběhne následovně. Z nově registrovaného zařízení (Device 1) se získá veřejný klíč Pk_1. Hashovací funkcí (např. SH-1) se z něj vytvoří hash e a ten je poslán do všech již zaregistrovaných zařízení (Device X). Pomocí stejných operací jako v Autentizační části se z hashe e vypočítá ověřovací zpráva z_x. Tato zpráva spolu s bodem $\bar{C}_x$ jsou odeslány zpět na zařízení Uživatel (Master device), které přičte ke všem (v případě více zařízení) bodům $\bar{C}_x$ stejným způsobem vypočítaný bod $\bar{C}_0$, čímž vznikne celkové $\bar{C}$. Všechny ověřovací zprávy $z_0 \ldots z_x$, celkový bod $\bar{C}$ a veřejný klíč nového zařízení jsou odeslány do Ověřovatele (Verifier). Stejnou hashovací funkcí vytvoří hash e' (hash e a e' by měly být totožné) a provede ověřující operaci

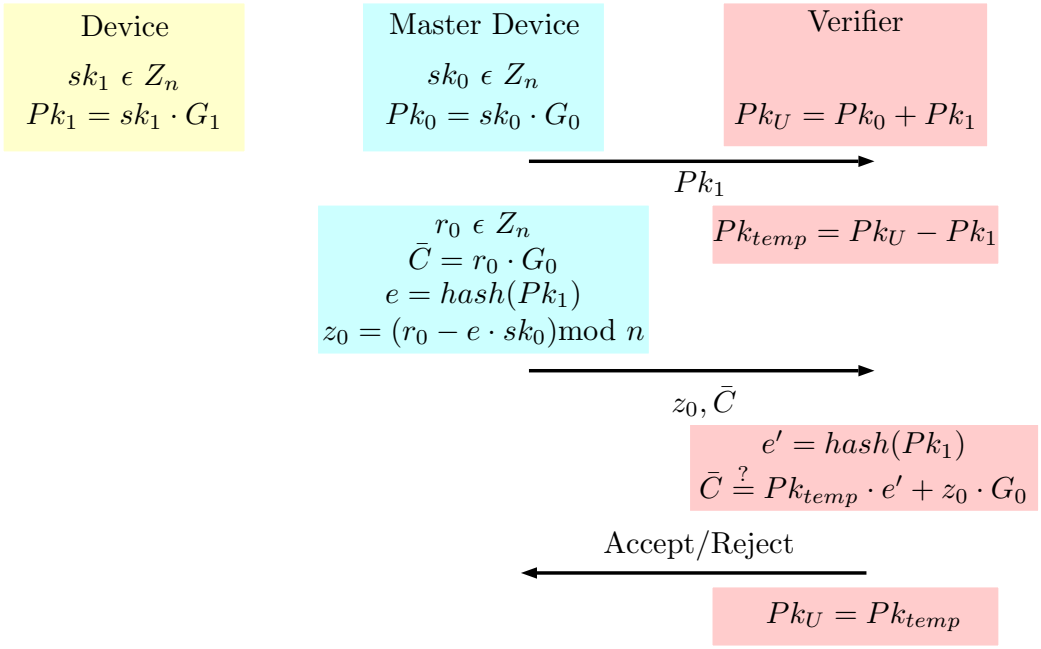
$$\bar{C} \stackrel{?}{=} Pk_U \cdot e' + z_0 \cdot G_0 + z_x \cdot G_x, \tag{2.5}$$

kde celkový veřejný klíč

$$Pk_U = sk_0 \cdot G_0 + sk_x \cdot G_x. \tag{2.6}$$

V posledním kroku Ověřovatel zhodnotí výsledek operací a vyhodnotí autentizaci, tedy i registraci. Pokud je ověření úspěšné, je k celkovému veřejnému klíči Pk_U přičten veřejný klíč nového zařízení Pk_1. K ověření identity se tedy použijí všechny již registrované entity a hlavní zařízení Master device (Uživatel). Výsledek registrace je odeslán na zařízení Uživatel a zde je nové zařízení uloženo jako registrované. Při

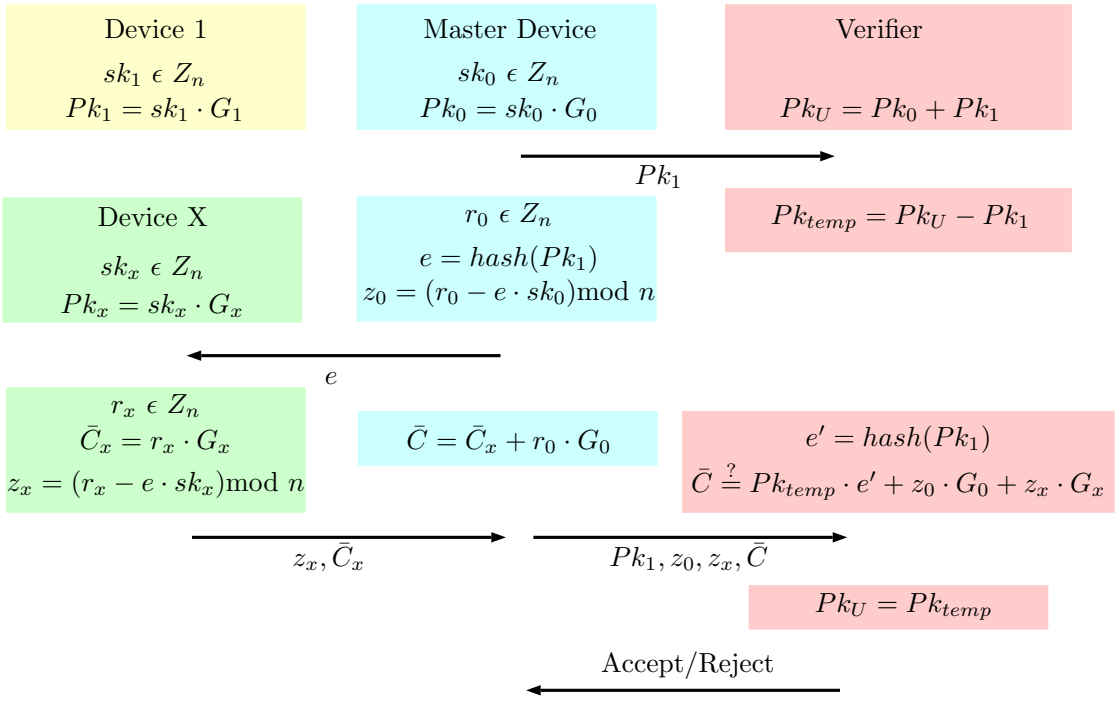
dalším procesu autentizace bude použito i toto zařízení. Bez něj by ověření (bez deregistrace) nebylo úspěšné.



Obr. 2.10: Deregistrační část – příklad deregistrace jediného zaregistrovaného zařízení.

Poslední částí protokolu HDM16 je deregistrační část. Jedná se o opak procesu registrace, kdy chceme nějaké již zaregistrované zařízení deregistrovat a v procesu autentizace ho nadále nepoužívat. Tento proces „smazání“ zaregistrovaného zařízení můžeme provést pouze pomocí znalosti jeho veřejného klíče Pk. Tento klíč je již z předešlého procesu registrace uložen v zařízení Uživatel a tudíž je deregistrace uskutečnitelná i bez fyzické přítomnosti tohoto zařízení. Tento proces je užitečný například v případě odcizení nebo ztráty jednoho ze zaregistrovaných zařízení.

Prvním případem je deregistrace jediného zaregistrovaného zařízení (obr. 2.10). Druhým případem je deregistrace zařízení při více jak jednom zaregistrovaném zařízení (obr. 2.11).



Obr. 2.11: Deregistrační část – příklad deregistrace při více zaregistrovaných zařízení.

Na straně hlavního zařízení Uživatel (Master device) se oba případy velmi podobají registraci, pouze se veřejný klíč právě deregistrovaného zařízení (Device 1) nezískává přímo od něj, jako při registraci, ale je uložen v databázi zaregistrovaných zařízení v Uživateli (Master Device). Další postup je již shodný s registrací. Na straně Ověřovatele se po obdržení klíče Pk_1 vypočítá dočasný klíč Pk_{temp} a ten se použije i v další procesu ověření identity. Pokud je autentizace úspěšná, je hlavní klíč Pk_U zaměněn za klíč Pk_{temp}, jinými slovy je od všech sečtených veřejných klíčů odečten klíč právě deregistrovaného zařízení Pk_1. Výsledek deregistrace je odeslán na hlavní zařízení Uživatel a ten v případě úspěchu zařízení odstraní z databáze zaregistrovaných zařízení a v dalším procesu autentizace již nebude použito.

3 TECHNOLOGIE NFC

Technologie NFC (Near Field Communication) je moderní technologie sloužící k bezdrátové komunikaci mezi elektronickými zařízeními na krátkou vzdálenost. Ta bývá řádově v jednotkách centimetrů (do 4 cm). Jedná se o radiovou komunikaci s vyhrazeným pásmem na frekvenci 13,56 MHz vycházející z technologie RFID (Radio Frequency Identification), se kterou uchovalo zpětnou kompatibilitu. Umožňuje sdílet data přenosovou rychlostí 106, 212 nebo 424 kb/s.

NFC se hodí především k přenosu menšího množství dat s nižším bezpečnostním rizikem. Komunikace je obousměrná, ale pouze v half-duplexním režimu, kdy jednotlivé zařízení může buď vysílat nebo příjmat, avšak nemůže oboje zároveň. V této kapitole bylo čerpáno z [10, 11, 12, 13].

Podle toho v jakém provozu zařízení komunikují, rozlišujeme:

- Aktivní NFC – obě komunikující strany se střídají v přenosu signálu.
- Pasivní NFC – jedna strana je napájena elektromagnetickým polem vysílače (např. NFC tag), takže nepotřebuje ke komunikaci žádné napájení.

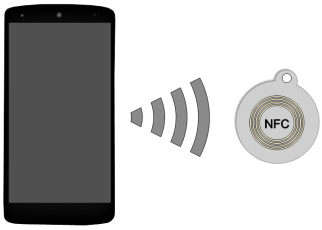
Od počátku vývoje se počítalo s nasazením především v mobilních telefonech a smart („chytrých“) zařízeních. Operační systém Android umožňuje podporu technologie NFC od verze Android 2.3, od verze Android 4.4 přibyla podpora emulace pasivního NFC, tedy možnost fungování v režimu emulace karet (viz níže).

3.1 Režimy NFC

Přenos s technologií NFC může fungovat ve třech režimech – v režimu pro zápis/čtení NFC tagů nebo čipových karet díky zprávám NDEF (Read/write režim), v režimu pro přenos dat mezi dvěma NFC aktivními zařízeními (Peer-to-peer) a v režimu emulace pasivního NFC prvku na aktivním zařízení (Režim emulace karet).

3.1.1 Read/write režim

Tento režim umožňuje zapisovat nebo číst tzv. NFC tagy nebo bezkontaktní čipové karty díky zprávám NDEF. Operační systém Android zpřístupnil tento režim jako první již od verze Android 2.3 Gingerbread (API level 9). Read/write režim je standardizován dle ISO/IEC 14443.



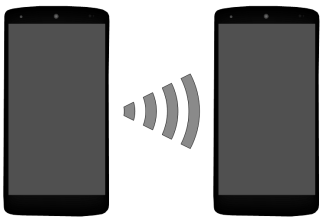
Obr. 3.1: NFC read/write režim

NFC Tag

NFC Tagy jsou malá zařízení fungující v pasivním NFC provozu. Pro komunikaci využívají energii naindukovanou elektromagnetickým polem iniciátora komunikace. Skládají se z čipu a antény (nejrozměrnější část zařízení). Důležitá je zde paměť zařízení [14].

3.1.2 Peer-to-peer režim

Jedná se o režim, ve kterém dochází k datovému přenosu mezi dvěma aktivními zařízeními s podporou NFC. Režim využívá funkce Android Beam pro přenos souborů (od verze Android 4.0). Maximální přenosové rychlosti u tohoto režimu dosahují až 424 kb/s. Peer-to-peer režim je standardizován dle ISO/IEC 18092.



Obr. 3.2: NFC Peer-to-peer režim

3.1.3 Režim emulace karet

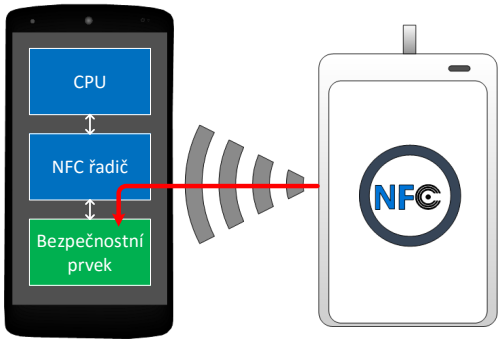
Režim emulace karet, anglicky také nazýván HCE (Host-based Card Emulation), umožňuje aktivnímu zařízení s podporou NFC chování jako pasivní NFC prvek (NFC tag, bezkontaktní čipová karta). Režim emulace čipové karty Android zpřístupnil teprve až od verze Android 4.4 KitKat (API level 19). Ke komunikaci se využívají zprávy APDU. Umožňuje autentizaci a autorizaci pouze pomocí mobilního zařízení, což lze využít např. pro bezkontaktní platby bez nutnosti fyzické přítomnosti platební karty. Využít jde ale například i pro přístupové systémy.

Obr. 3.3: NFC režim emulace karet

3.2 Režim emulace karet u zařízení s OS Android

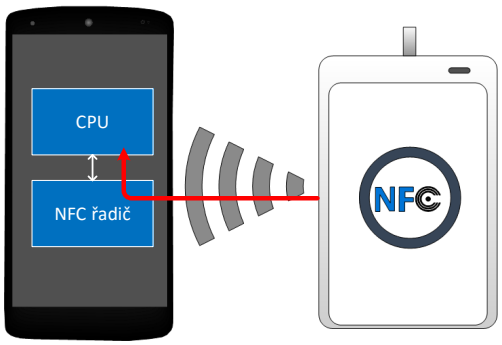
Režim emulace karet rozlišujeme podle použití bezpečnostního prvku.

Je-li využita emulace NFC karty za předpokladu použití bezpečnostního prvku (dále pouze jako „BP“), je při komunikaci čtečka navedena na BP umístěný na přístroji prostřednictvím ovládacího prvku NFC v zařízení. Komunikace tedy probíhá pouze mezi čtečkou a BP. Když uživatel drží zařízení nad NFC terminálem, posílá všechna data přímo na BP bez možnosti zasáhnutí jiné aplikace. Využití tohoto režimu má výhodu v tom, že ho mohou využít i zařízení s nižší verzí operačního systému Android [15]. Obrázek 3.4 demonstruje, jak komunikace probíhá.



Obr. 3.4: Emulace karet s použitím bezpečnostního prvku

Je-li využita emulace NFC karty bez použití bezpečnostního prvku, jsou data z čtečky posílána na CPU zařízení a aplikace mají možnost zasáhnout do komunikace. Aplikace na zařízení Android jsou spuštěny jako služby (services), probíhají tedy v pozadí aplikací. Při komunikaci jsou využívány zprávy APDU. Režim je umožněný zařízením s operačním systémem vyšší verze Android 4.4 [15]. Obrázek 3.5 demonstruje, jak komunikace probíhá.



Obr. 3.5: Emulace karet bez použití bezpečnostního prvku

Formát NDEF

Formát NDEF (NFC Data Exchange Format) je standardizovaný datový formát, který se používá pro výměnu informací mezi NFC zařízením a NFC tagem (případně druhým NFC zařízením). Formát je binárně strukturovaný do tzv. NDEF zpráv (messages), které se skládají z NDEF záznamů (records).

Obr. 3.6: Příklad NDEF zprávy

Každý záznam obsahuje záhlaví s informacemi o záznamu (např. ID, délka, TNF atd.) dlouhé 6–9 bajtů a payload (samotný obsah zprávy) proměnné délky. Jako příklad lze uvést zprávu určenou pro adresář s informacemi o kontaktu [16].

Zprávy APDU

Při emulaci bezkontaktních karet v režimu HCE využíváme ke komunikaci mezi zařízením a čtečkou datovou jednotku pro komunikaci s čipovými kartami APDU (Application Protocol Data Unit).

Při komunikaci uplatňujeme model master-slave, kdy čtečka karet vystupuje jako master a vysílá příkazy zprávou APDU Command. Zařízení emulující kartu jako slave na ně odpovídá zprávou APDU Response.

Zprávy jsou přenášeny pomocí protokolu TPDU (Transmission Protocol Data Unit). Datové jednotky APDU jsou standardizovány dle ISO 7816-4 [17, 18].

4 TECHNOLOGIE BLE

Technologie BLE (Bluetooth Low Energy, nazývaná také Bluetooth Smart) je bezdrátová PAN (Personal Area Network) komunikace představena v roce 2011 ve specifikaci Bluetooth 4.0 jako alternativa k Bluetooth Classic. Jak už ale název napovídá, technologie vznikla hlavně s cílem snížení energetické náročnosti. Použití je přínosné pro zařízení s malými energetickými prostředky, jakými jsou periferní zařízení.

Není příliš vhodný pro vzájemnou komunikaci telefonů (telefon → telefon), kde očekáváme vyšší nároky na rychlost, větší velikost přenášených dat a máme lepší možnosti čerpání energie. Mnohem vhodnější je pro periodické posílání menšího množství dat z periferního zařízení do centrálního prvku (např. ze senzoru srdečního tepu do telefonu). Uplatnění nachází díky rychle se rozvíjející myšlence IoT.

Pro zařízení s operačním systémem Android byla uvolněna podpora emulace pasivního prvku v centrálním módu od verze Android 4.3 (API Level 18). Periferní mód byl uvolněn až od Android verze 5.0 API 21 s nutnou hardwarovou podporou MLDP (Microchip Low-energy Data Profile) přenosu [19, 20, 21, 22].

4.1 Charakteristika technologie BLE

BLE, stejně jako jeho předchůdci, vysílá na radiových vlnách o frekvenci 2,4 GHz. Při přenosu využívá techniky adaptivního frekvenčního skákání, kdy zařízení vysílá část informace pouze na jednom kanále a ihned po odvysílání má možnost přeskočit na kmitočet jiného nosného, nerušeného kanálu. Díky složitému odposlechu je zajištěna vyšší bezpečnost a díky výběru nerušených kanálů i vyšší odolnost vůči radiovému rušení. Dosahuje přenosové rychlosti 1 Mb/s na fyzické vrstvě, nicméně jen kolem 230 kb/s na vrstvě aplikační, protože je přenos závislí na velkém množství nežádoucích faktorů. Nejkratší možná zpráva má délku 8 bajtů, nejdelší potom 27 bajtů. Pro detekci a opravu chyb během přenosu se využívá 24 bitový kontrolní součet CRC (Cyclic Redundancy Check).

Přestože BLE má k verzi Bluetooth Classic velmi blízko, kompatibilní s ní není. Proto pokud chceme zařízení, které bude schopné výměny informací s oběma stranami, musí toto zařízení ovládat obě tyto technologie [23].

Podle možnosti komunikace přes Bluetooth technologii rozlišujeme tři typy:

1. Bluetooth Classic – od verze Bluetooth 1.0, schopné komunikovat pouze se zařízeními technologie Classic Bluetooth (příp. její nadstavbou ERD s vyšší přenosovou rychlostí).
2. Bluetooth Smart – od verze Bluetooth 4.0, schopné komunikovat pouze se zařízeními technologie Bluetooth Smart, resp. Bluetooth Low Energy.

3. Bluetooth Smart Ready – od verze Bluetooth 4.0, zařízení sdružující obě předchozí technologie, schopné je tedy komunikovat se všemi zařízeními.

Obr. 4.1: Možnosti vzájemné komunikace zařízení přes technologii Bluetooth.

4.2 Generické profily v BLE

4.2.1 Generic Access Profile

GAP (Generic Access Profile) nám jednoznačně definuje, jak se zařízení podporující BLE mohou stát dostupné a jak spolu mohou dvě zařízení vzájemně komunikovat. Slouží nám k tomu dva mechanismy – mechanismus všesměrového vysílání (Broadcasting) a mechanismus spojení (Connecting) [23].

Mechanismus všesměrového vysílání

Rozlišujeme dvě role:

1. Všesměrový vysílač (Broadcaster) – jedná se o zařízení, které veřejně vysílá všesměrové pakety dat (advertisement).
2. Pozorovatel (Observer) – zařízení naslouchá všesměrovým paketům přijatých od vysílače. Žádná komunikace však zatím mezi vysílačem a pozorovatelem neprobíhá.

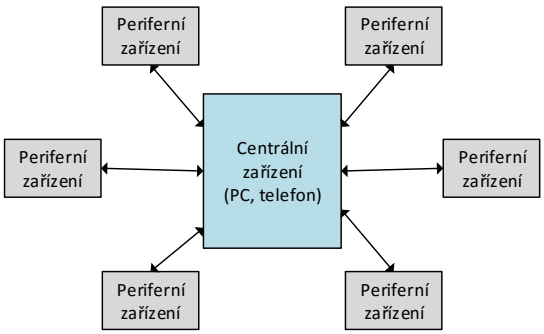
Tento mechanismus neslouží k navázání spojení z důvodu přenosu dat.

Mechanismus spojení

Rozlišujeme dvě role:

1. Periferní role – zařízení v periferním módu vysílá periodické data, aby oznámila svoji přítomnost a centrální prvek mohl navázat spojení. Pokud centrální prvek přijme „oznámení“ (advertisement), odešle na periferní zařízení žádost pro komunikaci a naváže se tak výhradní komunikace a periferní zařízení přestane „oznámení“ vysílat. Periferní prvek může být navázán komunikací pouze na jeden centrální (obr. 4.2). Spojení zůstává, dokud nedojde k rozpojení komunikace.
2. Centrální role – zařízení, které je v centrální roli opakovaně vyhledává zařízení a příjmá vysílaná data („oznámení“). Jsou-li data určena pro tento prvek

v centrální roli, iniciuje připojení. Jakmile je spojení navázáno, centrální prvek ovládá periodický přenos dat. Centrální prvek může mít libovolný počet připojených periferních zařízení.



Obr. 4.2: Topologie propojení centrálního a periferních zařízení.

Role v tomto mechanismu musí navázat výhradní komunikaci pro umožnění přenosu dat. Samo periferní zařízení komunikaci iniciovat neumí, stejně jako centrální prvek mnohdy nepodporuje funkci rozesílání oznámení, proto je pro navázání spojení nutné od každého prvku minimálně jeden.

4.2.2 Generic Attribute Profile

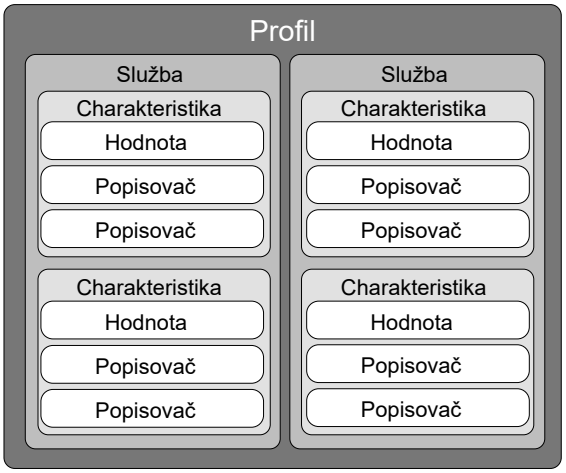
GATT (Generic Attribute Profile) je velmi podobný GAP, s tím rozdílem, že nám definuje komunikaci až po navázání spojení. Zaměřuje se hlavně na to, jak jsou data formátována, zabalena a poslána v souladu s určitými pravidly. Definuje nám, jak mezi sebou dvě BLE zařízení přenášejí data založená na konceptu charakteristik a služeb. Dělá tak s použitím obecných pravidel datového protokolu ATT (Attribute Protokol), který je použitý pro shromažďování služeb, charakteristik a ostatních souvisejících dat pomocí 16-bitových identifikátorů pro každou položku v jednoduché vyhledávací tabulce [23].

Rozlišujeme role:

1. Server – jedna z důležitých rolí je shromažďování a uchovávání atributů, na které se může GATT klient dotazovat.
2. Klient – typicky se jedná o zasílání požadavků na GATT server a očekávání odpovědi.

Rozdělení rolí je proměnné a záleží na tom, která strana si vyžádá (a tedy i příjmá) data a která je vysílá. Obyčejně se zařízení Android (telefon, tablet) chová jako

klient a vyžaduje si průběžná data od BLE periferního zařízení (chytré hodinky, náramky). Vysílající zařízení se chová jako server. Role se mohou obrátit například při vyžadování aktualizací BLE zařízením, které se nyní zachová jako klient a serverem se stane zařízení Android.



Obr. 4.3: Obrázek demonstruje jak Generic Attribute Profile vypadá

GATT přenos dat u BLE je založen na vnořených objektech nazývaných Profily, Služby a Charakteristiky. Hierarchie GATT je znázorněna na obr. 4.3.

- **Profil** – určitá skupina služeb na periferním zařízení. Profil je předdefinovaný buď od Bluetooth SIG (Special Interest Group) nebo přímo vývojáři periferních zařízení.

- **Služba** – rozdělují nám data do logických entit a obsahují tato specifická data jako tzv. Charakteristiky. Každá služba obsahuje jednu nebo více charakteristik. Pro vzájemné rozlišení služeb používáme jednoznačný celočíselný identifikátor zvaný UUID (Universally Unique IDentifier), který má buď 16 bitů (předdefinované BLE služby) nebo 128 bitů (vlastní nadefinované služby).

- **Charakteristika** – nachází se na nejnižší vrstvě konceptu GATT komunikace. Zapouzdřuje jednotlivá data, jako jsou například konkrétní hodnoty. Stejně jako služba i charakteristika používá buď předdefinovaný 16 nebo 128 bitový UUID k identifikaci. Některé charakteristiky jsou určeny pouze pro čtení (read-only), jiné mohou být přepsány. Hodnota charakteristiky může být následována tzv. Popisovačem (Descriptor). Popisovače jsou definované atributy, které popisují danou charakteristiku.

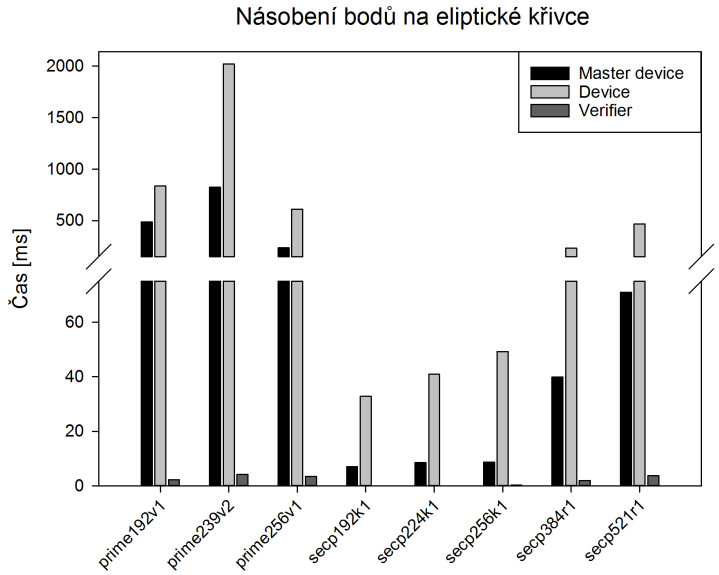
5 MĚŘENÍ KRYPTOGRAFICKÝCH PRIMITIV

V rámci této kapitoly bylo provedeno měření základních matematických operací využívaných v implementovaném protokolu HDM16. Měření bylo provedeno na těchto zařízeních: Chytré hodinky Sony SmartWatch 3 SWR50, telefon LG Nexus 5 s operačním systémem Android 5.1 Lollipop a počítač ACER TravelMate P253-E s operačním systémem Windows 10. Použitá byla knihovna Spongy Castle, což je upravená verze knihovny Bouncy Castle pro platformy s operačním systémem Android. Spongy Castle (resp. Bouncy Castle) je kolekce API používaných v kryptografii. Požitelná je jak pro programovací jazyk JAVA, tak i C#. U měření nebyly eliptické křivky (EC) náhodně generovány, ale pro kryptografii založené na eliptických křivkách nabízí knihovna spoustu předdefinovaných křivek. Pro měření byly využity tyto křivky: prime192v1, prime239v2, prime256v1, secp192k1, secp224k1, secp256k1, secp384r1, secp521r1. Křivky prime jsou definované organizací NIST (National Institute of Standards and Technology) [24], secp jsou pak křivky standardizované konsorciem SECG (Standards for Efficient Cryptography Group) [25]. Měřeny byly základní operace, které musí jednotlivé zařízení vykonat, pokud je autentizace založená na ECC (tabulka naměřených hodnot viz tab. 5.1).

Měřené operace:

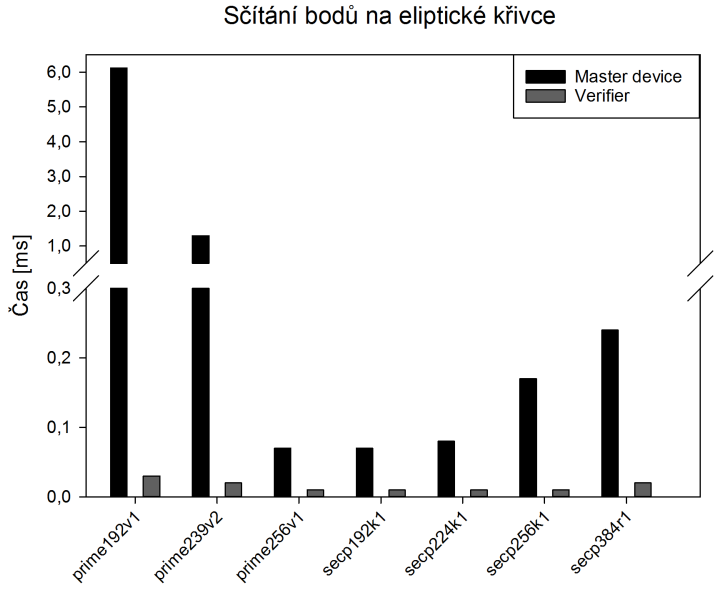
- KeyPair.gen – vygenerování páru klíčů pro konkrétní EC.
- RND – generování náhodného čísla bitové délky EC.
- PointMul – operace násobení bodů na EC.
- PointAdd – operace sčítání bodů na EC.
- Sub – odečítání dvou velkých čísel (proměnné BigInteger).
- modMul – modulární násobení dvou velkých čísel (proměnné BigInteger), modulus je dán řádem bodu eliptické křivky.
- mod – modulární redukce (zbytek po dělení) velkých čísel délky EC.

Změření každé operace bylo provedeno pro každou křivku stokrát, pro maximální přesnost a vyloučení náhodných výchylek, které mohli při měření vzniknout.



Obr. 5.1: Graf násobení bodů na křivce

Jak je vidět z výsledných časů i grafických znázornění křivky typu NIST měli značně delší časy například při operacích násobení (graf na obr. 5.1) a sčítání (graf na obr. 5.2) bodu na křivce. Zároveň si můžeme povšimnout výrazného rozdílu mezi časem výpočtu počítače (Verifier) s dostatečným výkonem, mobilním telefonem (Master Device) a hodinkami (Device).



Obr. 5.2: Graf sčítání bodů na křivce

Kromě výsledků je také uveden přepočítaný potřebný čas u role, která byla zařízení přidělena v souladu s protokolem HDM16. *Master time* je čas potřebný pro zařízení v roli Uživatele (v našem případě telefon). Pro tuto roli jsou potřebné operace RND, PointMul, PointAdd, Sub, modMul a mod. *Device time* je čas potřebný pro zařízení v roli Zařízení (v našem případě chytré hodinky). Pro roli Zařízení jsou potřebné operace RND, PointMul, Sub, modMul a mod. *Verifier time* je čas potřebný pro zařízení v roli Ověřovatele (v našem případě počítač). Pro Ověřovatele jsou potřebné operace RND, 2×PointMul a 3×PointAdd. Měření bylo kromě vlastního výpočetního výkonu zařízení závislé i na množství spuštěných aplikací a služeb v pozadí, proto jsou časy pro přibližné určení délky autentizace s použitím konkrétní EC čistě orientační.

Tab. 5.1: Tabulka naměřených hodnot kryptografických primitiv

Měření Eliptických křivek_Autentizace v IoT_PHONE								
EC	KeyPair.gen [ms]	RND [ms]	PointMul [ms]	PointAdd [ms]	Sub [ms]	modMul [ms]	mod [ms]	Master time [ms]
prime192v1	191,61	0,8	485,9	4,3	0,06	0,09	0,04	682,8
prime239v2	308,4	0,8	823,8	6,12	0,05	0,09	0,04	1139,3
prime256v1	86,2	0,9	235,2	1,3	0,05	0,09	0,04	323,8
Secp192k1	5,7	0,8	7,1	0,07	0,05	0,09	0,04	13,9
Secp224k1	6,7	0,8	8,6	0,07	0,05	0,09	0,05	16,4
Secp256k1	12,3	0,8	8,7	0,08	0,05	0,09	0,05	22,1
secp384r1	16,3	1,1	39,9	0,17	0,05	0,09	0,04	57,7
secp521r1	26,8	1,2	70,9	0,24	0,05	0,1	0,04	99,3

Měření Eliptických křivek_Autentizace v IoT_WEAR							
EC	KeyPair.gen [ms]	RND [ms]	PointMul [ms]	Sub [ms]	modMul [ms]	mod [ms]	Device time [ms]
prime192v1	370,3	0,3	836	0,07	0,3	0,05	1207,0
prime239v2	641,7	0,5	2022	0,07	0,1	0,05	2664,4
prime256v1	293,4	0,5	611	0,07	0,3	0,14	905,4
Secp192k1	36	0,3	32,8	0,07	0,1	0,05	69,3
Secp224k1	44,2	0,7	40,9	0,07	0,1	0,07	86,0
Secp256k1	52	0,5	49,2	0,07	0,1	0,07	101,9
secp384r1	104	0,9	233,3	0,07	0,1	0,05	338,4
secp521r1	142,2	0,9	468,3	0,07	0,2	0,05	611,7

Měření Eliptických křivek_Autentizace v IoT_PC					
EC	KeyPair.gen [ms]	RND [ms]	PointMul [ms]	PointAdd[ms]	Verifier time [ms]
prime192v1	10,9	0,02	2,24	0,02	17,7
prime239v2	3,36	0,01	4,2	0,03	16,0
prime256v1	3,36	0,01	3,4	0,02	13,6
Secp192k1	1,75	0,01	0,2	0,01	2,4
Secp224k1	1,44	0,01	0,2	0,01	2,1
Secp256k1	1,25	0,01	0,3	0,01	2,2
secp384r1	2,55	0,01	2	0,01	8,6
secp521r1	3,6	0,01	3,7	0,02	14,8

Time to Auth. [ms]	
prime192v1	1907,5
prime239v2	3819,8
prime256v1	1242,8
Secp192k1	85,6
Secp224k1	104,5
Secp256k1	126,2
secp384r1	404,7
secp521r1	725,8

6 IMPLEMENTACE PROTOKOLU HDM16

V rámci bakalářské práce byly prakticky implementovány tři části (resp. protokoly) již zmíněného protokolu HDM16, a to autentizační část, registrační část a nakonec deregistrační část. Vytvořeny byly tři aplikace pro PC, mobilní telefon s OS Android a chytré hodinky s OS Android. Projekt dostal pracovní název ECA (Elliptic Curve Authentication) a základní logo (obr. 6.1).

Obr. 6.1: Logo projektu ECA

6.1 Aplikace PC

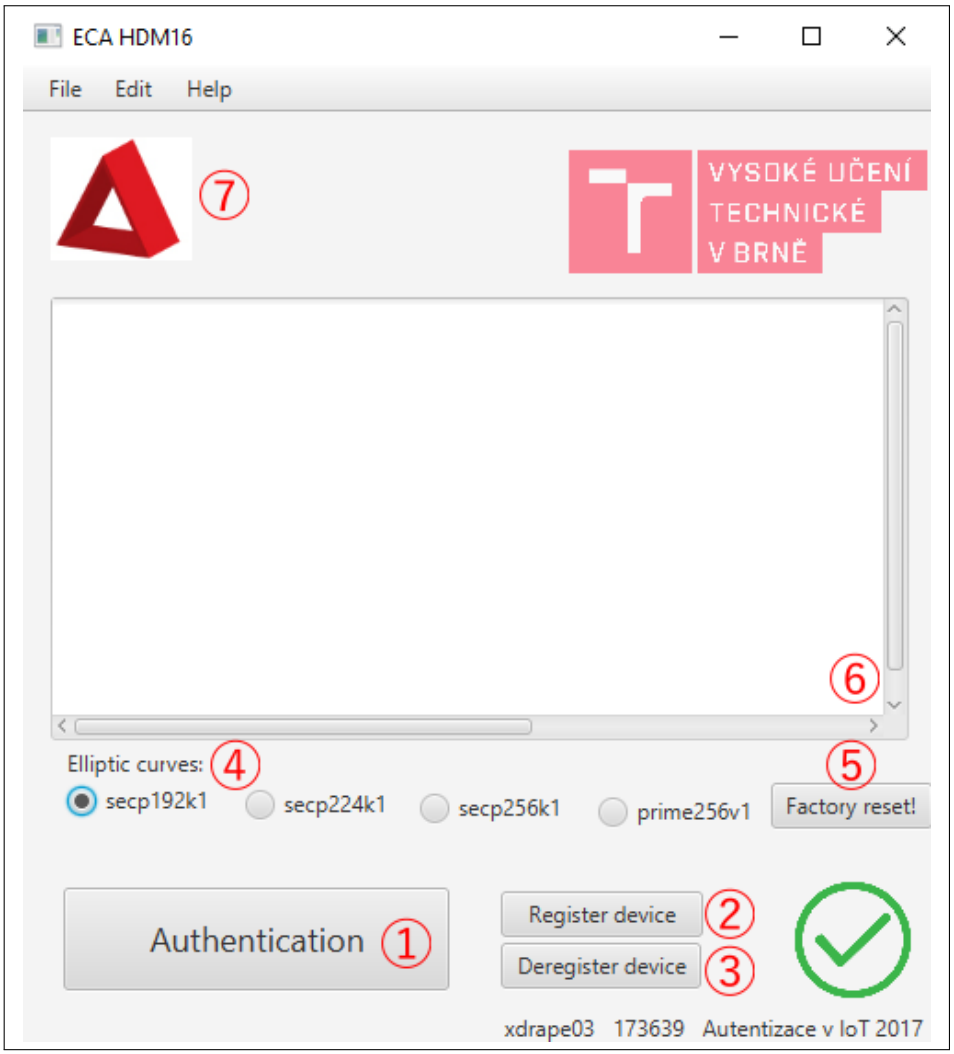
Aplikace pro počítač byla vytvořena ve vývojovém studiu Eclipse jako Java FX aplikace a grafické rozhraní bylo vytvořeno pomocí prostředí Scene Builder. Do prostředí byly naimportovány knihovny pro práci se čtečkou bezdrátových karet (smartcardio etc.) a dvě knihovny z kolekce API pro kryptografii Spongy Castle (core-1.54.0.0 a prov-1.54.0.0) pro operace nad eliptickými křivkami.

6.1.1 Grafické rozhraní PC

Ukázka základního grafického rozhraní je na obr. 6.2. Použity byly čtyři základní grafické prvky:

- Button (na obrázku popisek č. 1, 2, 3 a 5)
- TextArea (na obrázku popisek č. 6)
- RadioButton (na obrázku popisek č. 4)
- ImageView (na obrázku popisek č. 7)

Tlačítko s názvem Authentication (č. 1) je umístěno v dolní části grafického rozhraní. Po jeho zmáčknutí dojde ke spuštění celého procesu Autentizace a PC očekává přiložení mobilního telefonu k terminálu. Proces je možný provést hned po spuštění aplikace, ale pokud není zaregistrované žádné zařízení (resp. pokud není žádné zařízení uloženo v databázi zaregistrovaných zařízení), použije se pro ověření pouze mobilní telefon.



Obr. 6.2: Grafické rozhraní - aplikace PC

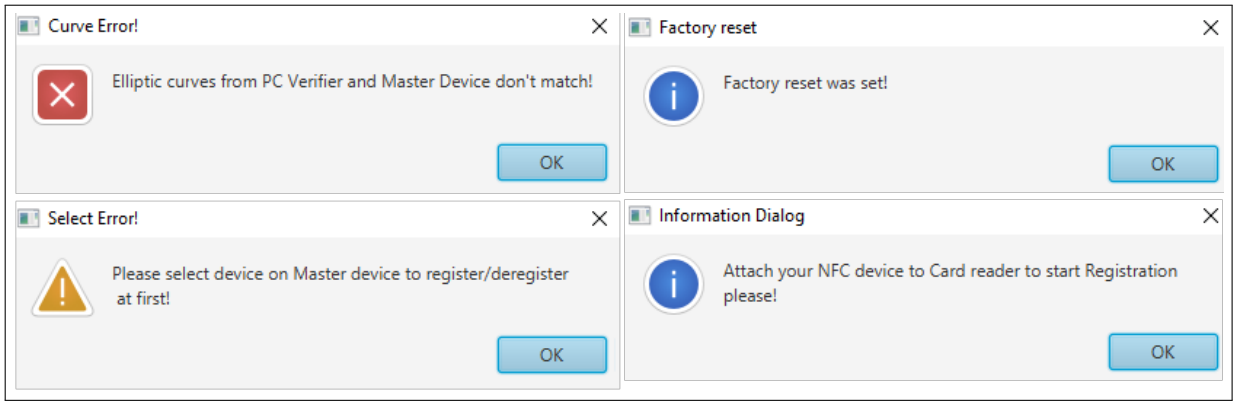
Tlačítka Register device (č. 2) a Deregister device (č. 3) slouží k zaregistrování nebo odregistrování určitého zařízení do nebo z databáze zaregistrovaných zařízení. Před spuštěním jednoho z těchto dvou procesů je nutné vybrat konkrétní zařízení v grafickém rozhraní mobilního telefonu. Pokud vybráno není, objeví se tzv. Alert dialog, který na to uživatele upozorní. Vedle tlačítek se nachází indikátor úspěšnosti operací, na tomto obrázku se konkrétně jedná o indikátor úspěšné autentizace.

RadioButtony (č. 4) slouží k výběru eliptické křivky, která bude pro následující proces použita. Tlačítka jsou seskupena do tzv. ToggleGroup, takže je zabezpečeno, že bude vybraná pouze jedna volba.

Tlačítko s názvem Factory reset! (č. 5) bylo přidáno z potřeby resetu aplikace do původního nastavení, kdy se smažou všechna zaregistrovaná zařízení. Po stisknutí se objeví přihlašovací dialog a před samotným „vyresetováním“ musí uživatel prokázat svoji identifikaci pomocí jména a hesla.

Do pole TextArea (č. 6) se vypisují informace o právě probíhajícím spuštěném procesu (např. použitá křivka, hodnoty proměnných, výsledek procesu, uběhlý čas atd.). Při inicializaci se zkontroluje, zda je k počítači připojena čtečka čipových karet. Pokud je všechno v pořádku načte se do proměnné typu CardTerminal a dále se pracuje s tímto terminálem. Vypíše se text „Choose the Elliptic curve and push 'Authentication'“ a aplikace je připravena k použití.

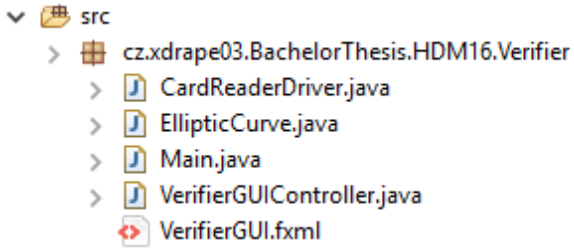
V záhlaví aplikace se zobrazují dva prvky typu Image View (č. 7). Jedná se o logo aplikace (vlevo) a logo školy (vpravo). Slouží pouze pro vizuální vzhled aplikace.



Obr. 6.3: Grafické rozhraní PC - Alert dialogy

Jak již bylo řečeno, protože může nastat neočekávané chyba v komunikaci mezi PC a telefonem, jsou doplněny protokoly o zprávy, které se přenáší v případě nesouladu a zobrazí se Alert dialog s konkrétní informací nebo upozorněním (např. při nepoužití stejné eliptické křivky na obou stranách, nevybrání zařízení pro registraci/deregistraci na telefonu nebo pouze informativní dialogy o provedení operace). Příklady Alert dialogů jsou na obr. 6.3.

6.1.2 Struktura a vývoj



Obr. 6.4: Struktura projektu - aplikace počítače

Struktura projetku se skládá ze 4 tříd a jednoho fxml souboru (viz obr. 6.4).

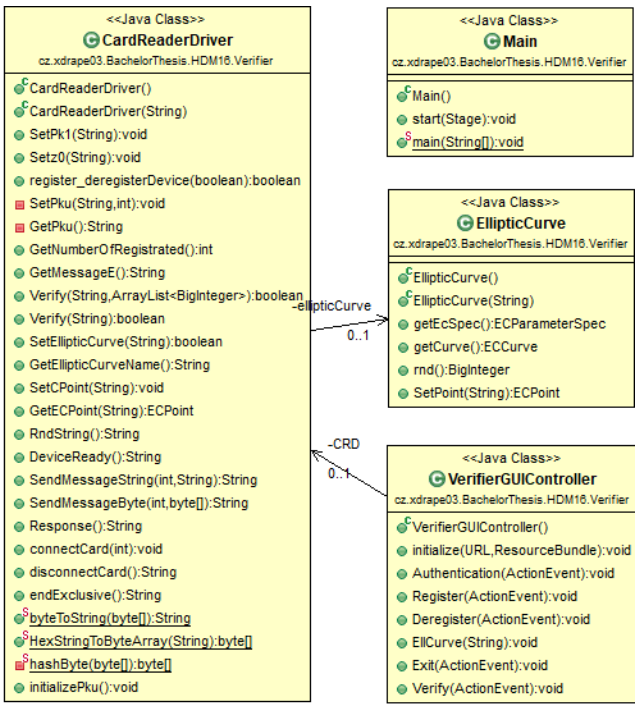
- Main – Jedná se především o spouštěcí třídu. Je zde definovaný fxml spouštěcí soubor a nastavení scény.

- EllipticCurve – Tato třída umožňuje vytvářet objekty typu ECCurve pomocí názvu předdefinované křivky a nabízí další přídavné operace nad eliptickými křivkami.

- CardReaderDriver – Třída slouží především jako pomocná, která drží všechny důležité metody jako je metoda ověření identity, registrace atd. Dále také umožňuje komunikaci přes čtečku čipových karet pomocí rámců APDU.

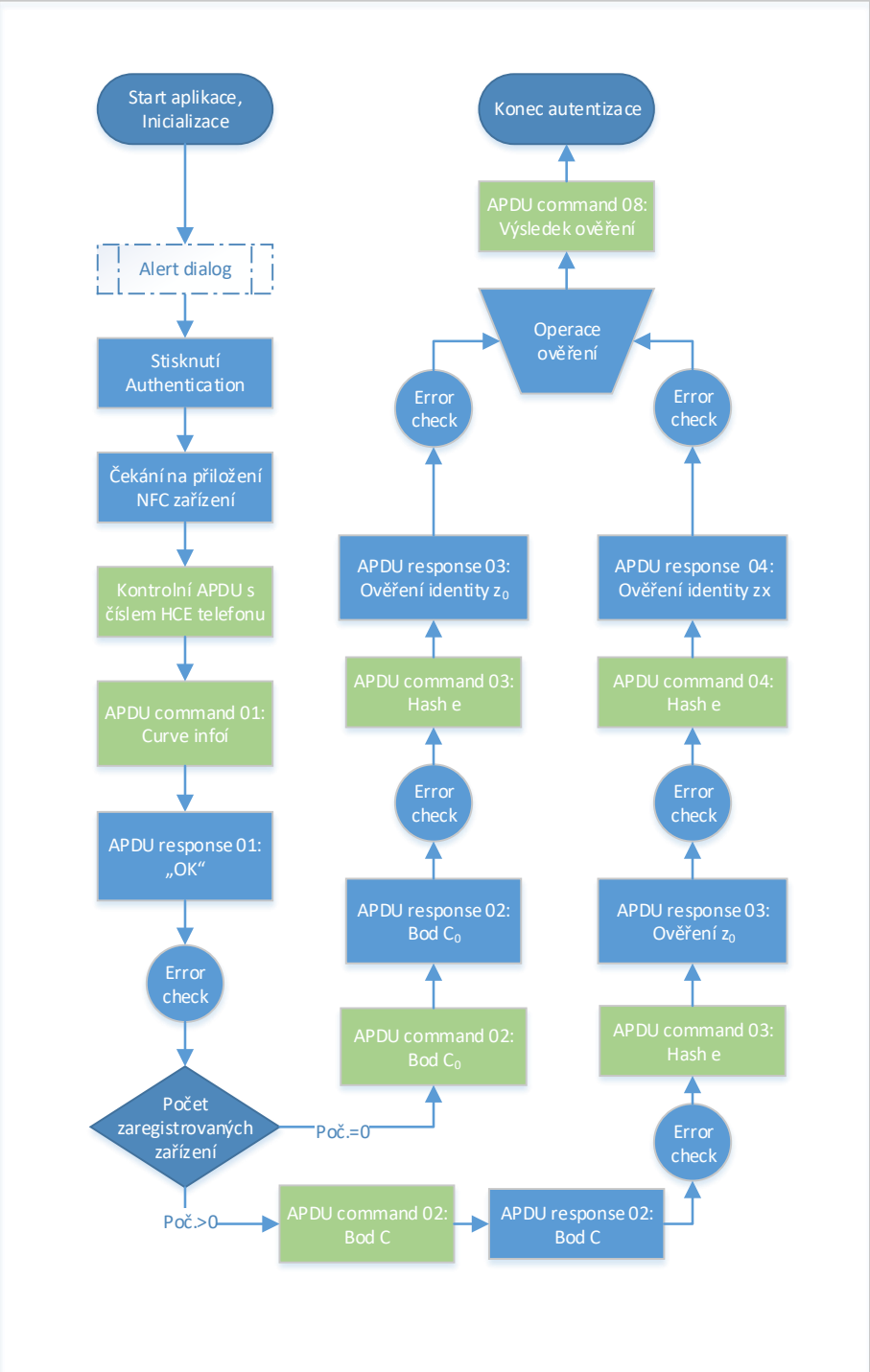
- VerifierGUIController – Tato třída je svázaná s grafickými ovládacími prvky ze souboru fxml. Je to hlavní řídící třída, která pracuje především s metodami třídy CardReaderDriver.

- VerifierGUI.fxml – Grafický spouštěcí soubor, který drží všechny grafické prvky aplikace. Vytvořený byl pomocí programu Scene Builder. Vzhled souboru viz obr. 6.2.



Obr. 6.5: Struktura projektu PC - UML diagram

Obrázek 6.5 zobrazuje UML diagram PC aplikace. Můžeme si prohlédnout vztah tříd a definované metody. Kontrolní třída VerifierGUIController drží objekt typu CardReaderDriver (CRD), ten opět pracuje s objektem třídy EllipticCurve (ellipticCurve).



Obr. 6.6: Digram autentizace - aplikace PC

Na obrázku 6.6 si můžeme prohlédnout zjednodušené schéma jak je počítač zapojen do autentizačního procesu. Je zřejmé, že řízení celého procesu leží právě na této aplikaci. Operace „Error check“ slouží k zajištění správného chodu procesů. Jedná se o přijaté APDU s definovanou chybovou zprávou. Pokud je obdržena objeví se Alert dialog a proces se přeruší. Z každého bloku „Error check“ směřuje pomyslná cesta na blok „Alert dialog“.

6.2 Aplikace mobilní telefon

Aplikace pro telefon byla vytvořena ve vývojovém studiu Android Studio. Kvůli operacím nad eliptickými křivkami byly opět přidány knihovny z kolekce API pro kryptografii Spongy Castle (core-1.54.0.0 a prov-1.54.0.0).

6.2.1 Grafické rozhraní mobilní aplikace

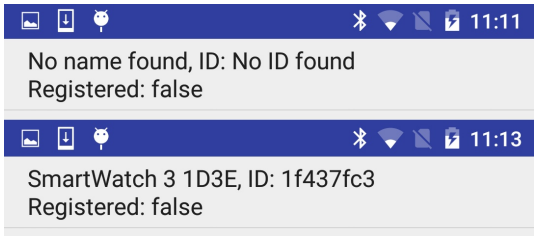
Ukázka základního grafického rozhraní mobilní aplikace je na obr. 6.7. Použity byly čtyři základní grafické prvky:

- Button (na obrázku popisek č. 1, 2, 3 a 4)
- Spinner (na obrázku popisek č. 5)
- TextView (na obrázku popisek č. 6)
- ImageView (na obrázku popisek č. 7)

Obr. 6.7: Grafické rozhraní - aplikace mobilní telefon

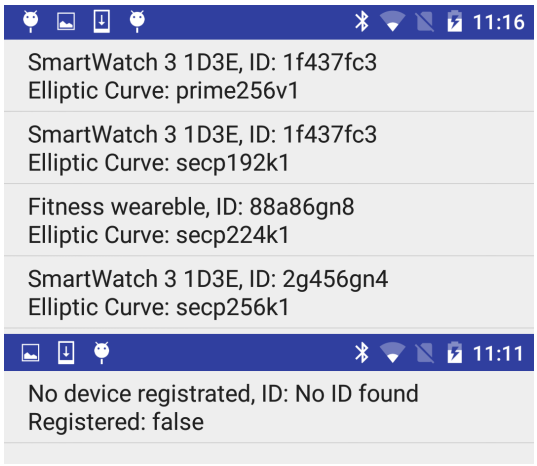
Tlačítko s názvem PUT VIRTUAL (č. 1) slouží pouze k vývojovým účelům. Jedná se o umělé přidání dvou zařízení do databáze zaregistrovaných zařízení, která ale nemohou být k autentizaci použita, protože neproběhl proces registrace společně s PC.

Tlačítko FACTORY RESET (č. 2) bylo přidáno ze stejného důvodu jako na PC. Slouží k resetu aplikace do původního nastavení, kdy se smažou všechna zaregistrovaná zařízení z databáze.



Obr. 6.8: Grafické rozhraní - Aktivita zobrazující dostupná zařízení

Tlačítko s názvem AVAILABLE DEVICES (č. 3) spustí aktivitu (viz obr. 6.8), zobrazí všechna dostupná zařízení a její status registrace pro právě zvolenou eliptickou křivku (případ dole na obr. 6.8). Případně zobrazí, že nejsou žádná zařízení dostupná (případ nahoře na obr. 6.8).

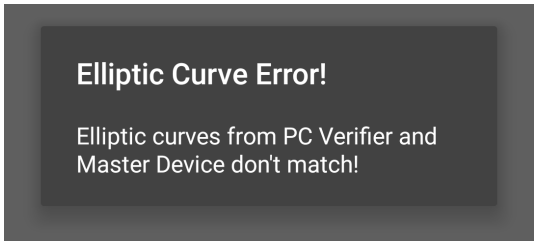


Obr. 6.9: Grafické rozhraní - Aktivita zobrazující zaregistrovaná zařízení

Tlačítko VIEW REGISTERED DEVICES (č. 4) spustí aktivitu (viz obr. 6.9) a zobrazí všechna zaregistrovaná zařízení a eliptickou křivku, pro kterou je dané zařízení registrováno (případ nahoře na obr. 6.9). Případně zobrazí, že nejsou žádná zařízení zaregistrovaná (případ dole na obr. 6.9).

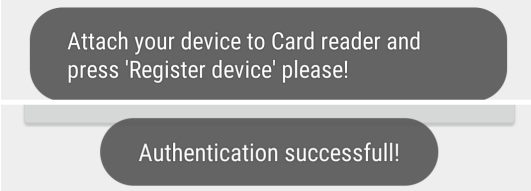
Spinner (č. 5) a popisek „Elliptic curve:“ (č. 6) slouží k výběru právě používané eliptické křivky. Od tohoto výběru se odvíjí, jaká zařízení z databáze zaregistrovaných budou použita a v případě registrace, k jaké křivce budou přiřazena.

V záhlaví se opět zobrazuje logo aplikace (č. 7). Slouží pouze k vizuálnímu vzhledu aplikace.



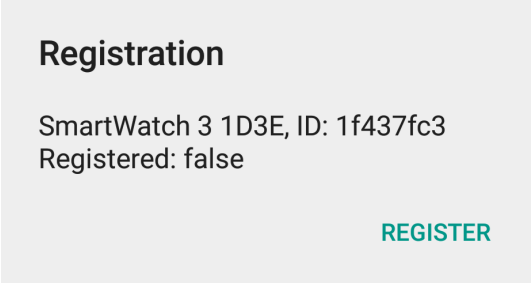
Obr. 6.10: Grafické rozhraní mobilního telefonu - Alert dilog

V případě nějakého problému je aplikace, stejně jako aplikace na PC, vybavena Alert dialogy pro upozornění uživatele o nějakém problému. Příklad dialogu je na obr. 6.10.



Obr. 6.11: Grafické rozhraní mobilního telefonu - Toasty

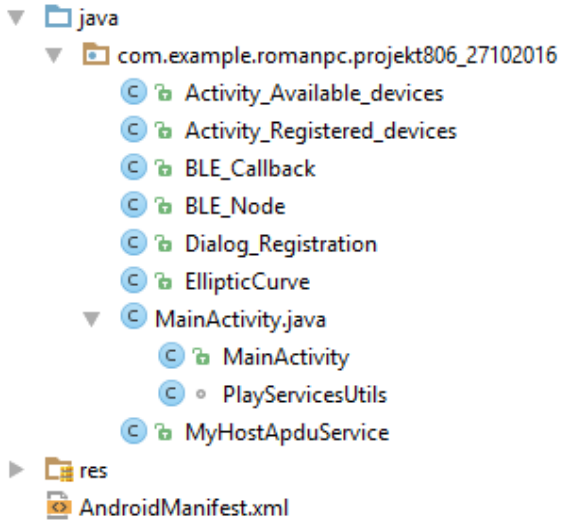
Pro komunikaci s uživatelem, například o úspěchu/neúspěchu autentizace, registrace atd. jsou použity tzv. Toasty. Příklady toastu jsou na obrázku 6.11.



Obr. 6.12: Grafické rozhraní mobilního telefonu - Dialog registrace

V případě aktivity VIEW REGISTERED DEVICES i AVAILABLE DEVICES je možné měnit statut registrace zařízení. Po kliknutí na konkrétní zařízení je zobrazen dialog (obr. 6.12) s možností buď registrace nebo deregistrace, podle aktuálního stavu zařízení. V případě zobrazených dostupných zařízení je možnost obou voleb (registrace i deregistrace). Při zobrazených již zaregistrovaných zařízení je možnost pouze deregistrace (pro možnost deregistrace i v případě nedostupného zařízení).

6.2.2 Struktura a vývoj



Obr. 6.13: Struktura projektu - mobilní aplikace

Struktura projetku se skládá z 10 tříd, z toho tři aktivity a jeden dialog (obr. 6.13).

- MainActivity – Hlavní základní a domácí aktivita. Nabízí všechny důležité grafické prvky a zajišťuje jejich propojení s ovládacími prvky. Svázána s grafickým xml souborem activity_main.xml (viz obr. 6.7).

- PlayServicesUtils – Pomocná třída pro zjištění aktuální verze GooglePlay API a možnosti upgradu (pro správnou funkci Wearable API).

- EllipticCurve – Stejně jak na PC umožňuje definovat objekt ECCurve pomocí názvu křivky a nabízí další pomocné metody a operace nad eliptickými křivkami.

- Dialog_Registration – Třída rozšiřuje třídu DialogFragment. Slouží jako registrační dialog pro možnost změny statusu registrace určitého zařízení (viz obr. 6.12).

- Activity_Available_devices – Jedna z vedlejších aktivit zobrazující všechna dostupná zařízení. Po kliknutí na určitou položku v seznamu (dostupné zařízení) zobrazí dialog třídy Dialog_Registration. Aktivita je svázaná s xml souborem activity_register_activity, který udává její vzhled (viz obr. 6.8).

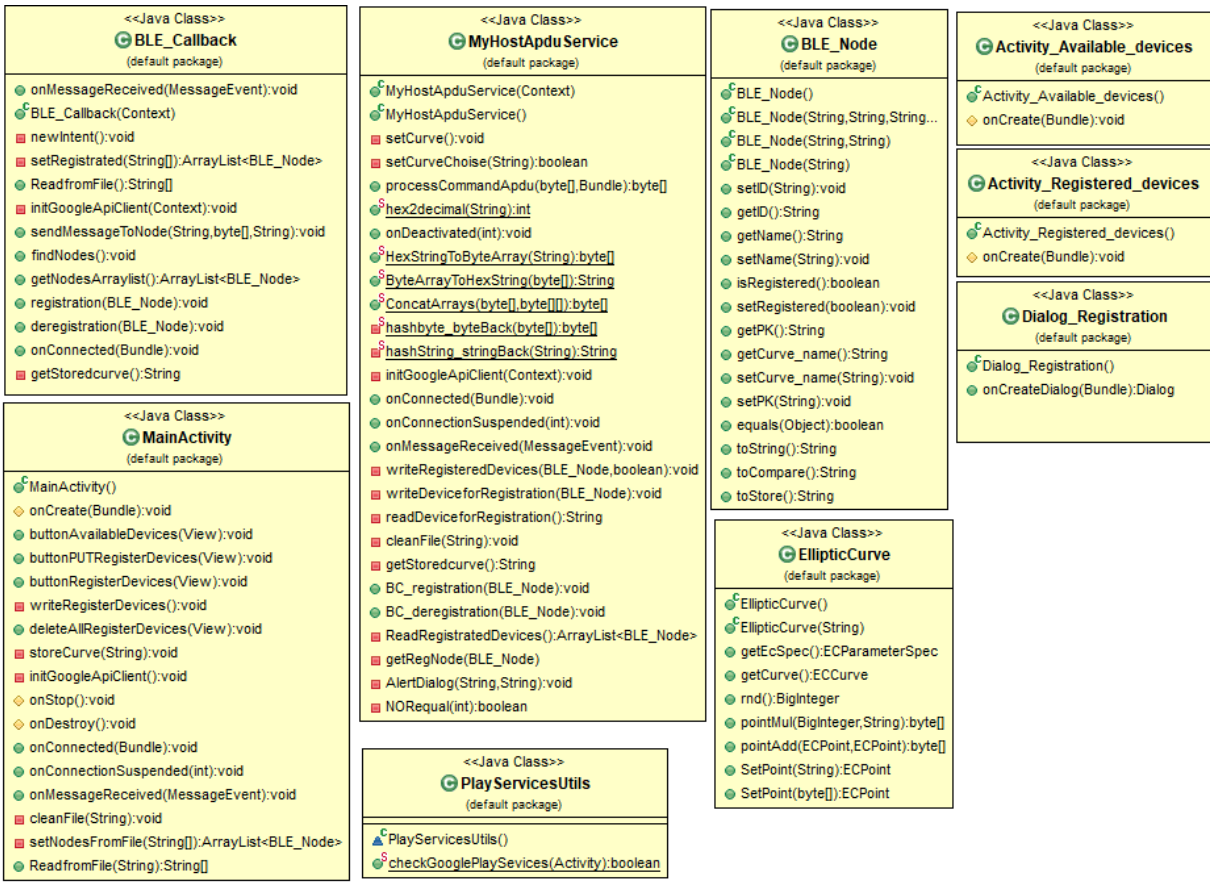
- Activity_Registered_devices – Druhá vedlejší aktivita zobrazující všechna zaregistrovaná zařízení vyčtená ze souboru. Stejně jako aktivita dostupných zařízení, i tato po kliknutí na určitou položku (již zaregistrované zařízení) zobrazí dialog třídy Dialog_Registration. Svázaná je se stejným xml souborem activity_register_activity, který umožňuje vzhled typu seznam (viz obr. 6.9).

- BLE_Node – Všechna zařízení (BLE uzly), se kterými program pracuje, převádí na objekty této třídy. Každý objekt má určité vlastnosti jako je ID (jednoznačný identifikátor zařízení), name (název zařízení), isRegistered (status registrace), Curve_name (jméno křivky pro kterou je zaregistrováno v případě zaregistrovaného zařízení) a Pk (veřejný klíč, také pouze v případě zaregistrovaného zařízení).

- BLE_Callback – Tato třída umožňuje komunikaci přes BLE – např. posílat zpětnou vazbu na konkrétní zařízení (dle ID) nebo vyhledávat dostupná zařízení.

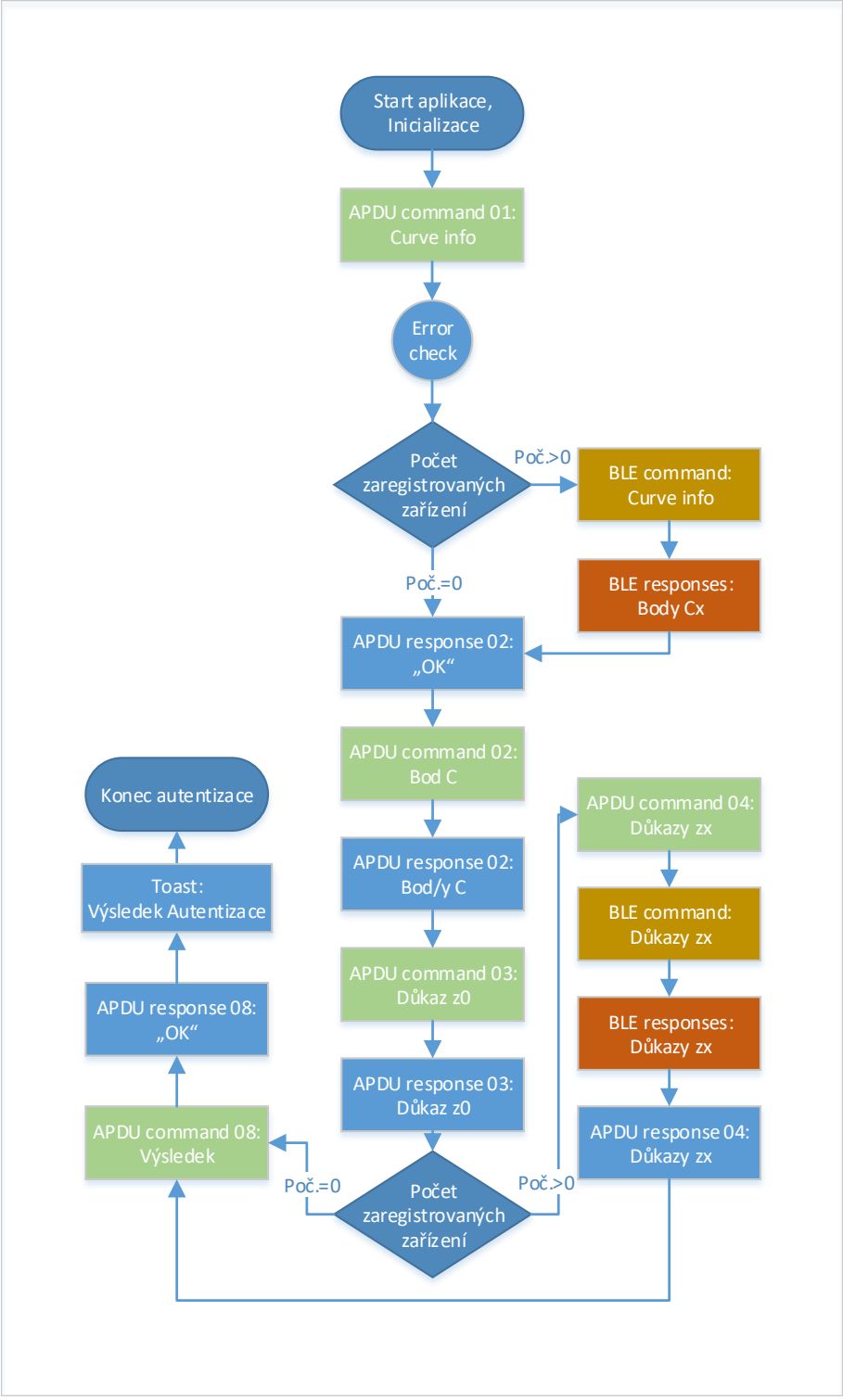
- MyHostApduService – Třída je spuštěna jako služba na pozadí. Odpovídá na dotazy APDU zpráv obdržených od čtečky a stará se o hlavní řízení procesů autentizace, registrace atd. Třída také přijímá zprávy přijaté přes BLE.

- AdroidManifest.xml – Velice důležitá část každého Android projektu. Specifikuje všechna svolení (permissions), kterými aplikace disponuje. Tato aplikace má svolení k používání NFC, zápisu a čtení z paměti, možnosti WAKE_LOCK (udržení procesoru mimo spánek nebo obrazovky před zhasnutím) a nakonec možnosti zobrazovat Alert okna.



Obr. 6.14: UML Digram - aplikace telefon

Obrázek 6.14 zobrazuje zjednodušený UML diagram aplikace na telefon. Můžeme si všimnout, že vzhledem k ostatním aplikacím se jedná o nejsložitější strukturu programu. Je to tak jednak kvůli složitějšímu grafickému rozhraní, více aktivitám a také kvůli potřebě komunikovat jak s PC čtečkou karet, tak i hodinkami. Implementovány proto musí být všechny prvky umožňující komunikaci skrze tyto dvě technologie. Třída BLE_Callback obsahuje ještě mimo zobrazené metody abstraktní třídy Context.



Obr. 6.15: Digram autentizace - aplikace PC

Na obrázku 6.15 si můžeme prohlédnout zjednodušené schéma jak je mobilní telefon zapojen do autentizačního procesu. Jak si můžeme povšimnou, telefon se zde stává prostředníkem mezi PC a hodinkami (resp. všemi periferiemi).

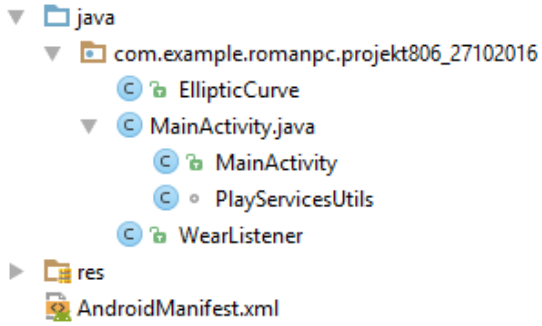
Operace „Error check“ slouží ke zjištění, zda je zvolená stejná eliptická křivka a zda souhlasí počet zaregistrovaných zařízení. Bloky začínající „APDU“ značí komunikaci se čtečkou, bloky začínající „BLE“ komunikaci s BLE uzly.
Díky tomu, že aplikace využívá služby spuštěné na pozadí a ukládání do souboru, je možné uskutečnění autentizace i bez nutnosti spuštění aplikace. Postačí pouze odemčení z úsporného módu (rozsvícení uzamčené obrazovky).

6.3 Aplikace smartwatch

Aplikace pro chytré hodinky byla také vytvořena ve vývojovém studiu Android Studio. Kvůli operacím nad eliptickými křivkami byly opět přidány knihovny z kolekce API pro kryptografii Spongy Castle (core-1.54.0.0 a prov-1.54.0.0).

Grafické rozhraní je tentokrát tvořeno pouze logem aplikace, protože aplikace běží především jako služba na pozadí a není potřeba žádných uživatelských operací.

6.3.1 Struktura a vývoj



Obr. 6.16: Struktura projektu - aplikace na hodinkách

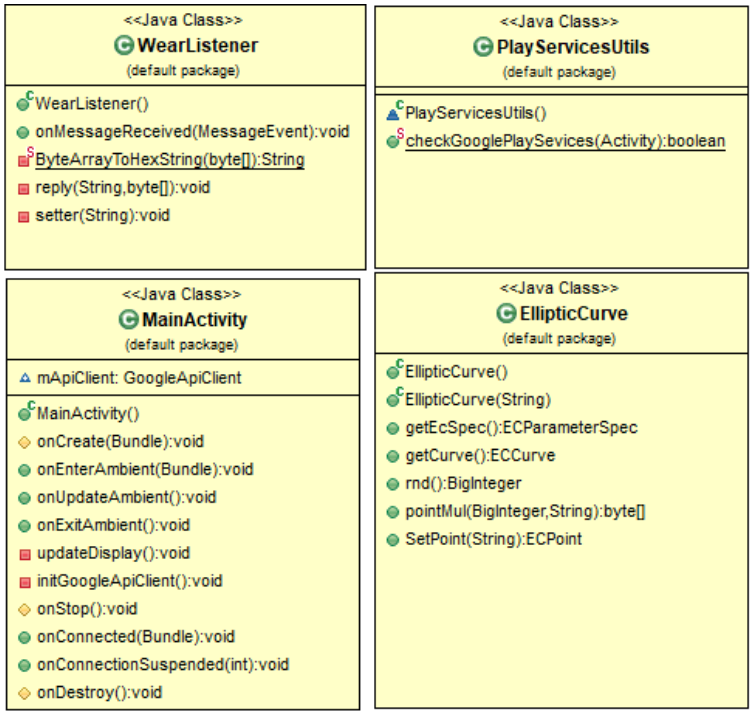
Struktura projetku se skládá ze 4 tříd (viz obr. 6.16).

- MainActivity – Základní aktivita. Obsahuje také konktrolní třídu PlayServicesUtils pro zajištění aktulní verze GooglePlay API. Svázána s grafickým xml souborem activity_main.xml (obsahuje pouze logo aplikace).

- EllipticCurve – Tato třída umožňuje vytvářet objekty typu ECCurve pomocí názvu předdefinované křivky a nabízí další přídavné operace nad eliptickými křivkami.

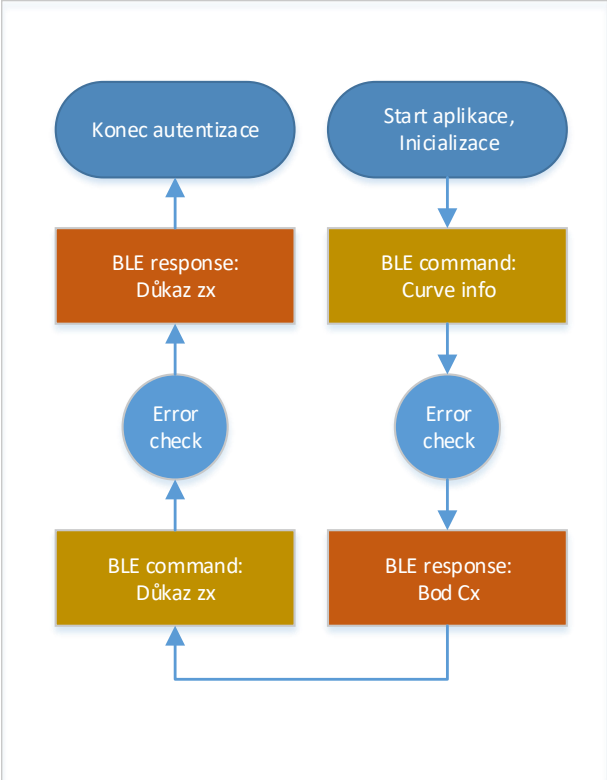
- WearListener – Rozšiřuje třídu WearableListenerService. Je spuštěná jako služba na pozadí. Plní funkci řídící třídy všech procesů.

- AdroidManifest.xml – Specifikuje všechna svolení (permissions), kterými aplikace disponuje. Tato aplikace má svolení pouze k odebírání zpráv BLE Message a používání možnosti WAKE_LOCK (udržení procesoru mimo spánek nebo obrazovky před zhasnutím).



Obr. 6.17: UML Digram - aplikace telefon

Obrázek 6.17 zobrazuje zjednodušený UML diagram aplikace na hodinkách. Můžeme si všimnout, že vzhledem k ostatním aplikacím se jedná o nejjednodužší strukturu programu. Je tak učiněno z důvodu minimálního zatížení hodinek, protože nedisponuje takovým výkonem a z důvodu úspory energie.



Obr. 6.18: Digram autentizace - aplikace SmartWatch

Na obrázku 6.18 si můžeme prohlédnout zjednodušené schéma jak jsou hodinky zapojené do autentizačního procesu. Můžeme si povšimnout, že na hodinky jsou kladeny minimální nároky. Před každým odesláním odpovědi „BLE response“ je vypočítán požadavek. Ostatní operace provádějí PC a telefon.

6.4 Experimentální testování a měření

V rámci Bakalářské práce bylo provedeno i praktické měření průměrného času potřebného k autentizaci. Měření bylo prováděno ve dvou variantách: Autentizace pouze telefonem a telefonem se zaregistrovanými hodinkami (tabulka naměřených hodnot viz tab. 6.1). Měření bylo provedeno nad čtyřmi standardizovanými EC, které jsou již předdefinované v knihovně Spongy Castle. Jedná se o křivky: secp192k1, secp224k1, secp256k1 a prime256v1.

Měření bylo provedeno na těchto zařízeních: Chytré hodinky Sony SmartWatch 3 SWR50, telefon LG Nexus 5 s operačním systémem Android 5.1 Lollipop a počítač ACER TravelMate P253-E s operačním systémem Windows 10.

Komunikace mezi hodinkami a telefonem probíhala pomocí BLE – Bluetooth Low Energy (využívalo se MessageApi). Počítač komunikoval pomocí čtečky čipových karet (ACR122U USB NFC Reader). Komunikace mezi telefonem a počítačem

Tab. 6.1: Praktické měření času potřebného k autentizaci.

Čas *t* potřebný k autentizaci - Pouze telefon				
Elliptická křivka	secp192k1	secp224k1	secp256k1	prime256v1
Průměr t [ms]	210,9	236,9	255,5	542,2
Směrodatná odchylka	18,3	27,4	15,8	79,5
Čas *t* potřebný k autentizaci - Telefon i hodinky				
Průměr t [ms]	805,5	1064,7	1157,5	1891,4
Směrodatná odchylka	43,3	197,6	205,6	175,9

probíhala technologií NFC – Near Field Communication (rámce APDU).

Veřejné klíče Pk_x jsou v telefonu uloženy v souboru ()zároveň se jménem a ID uzlu, ke kterému náleží. Měření bylo ovlivněnu různými faktory. Například výkonem zařízení, spuštěnými službami na pozadí obou zařízení a tentokrát také dobou komunikace jednotlivých zařízení. Nejdelší čas komunikace byl zaznamenán mezi hodinkami a telefonem, což se výrazně promítlo i ve výsledky měření.

7 ZÁVĚR

Bakalářská práce se zabývá možnostmi autentizace v IoT. V úvodu práce byly popsány některé vybrané autentizační protokoly z oblasti IoT. Jejich přednostmi je redukce potřebných úkonů, které dokážou zajistit jednoznačnou identifikaci zařízení s nízkým výpočetním výkonem. Protokoly často využívají třetí stranu (certifikační autoritu), která zajišťuje nezaměnitelné certifikáty pro danou entitu.

Dále byly popsány technologie NFC a BLE, které nacházejí uplatnění v oblasti IoT díky své energetické nenáročnosti. Tyto technologie byly využity v praktické části práce.

V rámci práce byly také proměřeny operace nad různými typy eliptických křivek a modulární aritmetiky potřebné pro autentizaci dle schématu protokolu HDM16. Pro křivky typu NIST vychází celková doba protokolu přibližně 1,2–3,8 s a pro křivky typu SECG vychází celkový čas přibližně 86–726 ms.

Výstupem jsou aplikace pro tři platformy (PC, mobilní telefon a SmartWatch), které mají implementované tři protokoly (autentizační, registrační a deregistrační část). Protokoly jsou v souladu s Multi-device protokolem HDM16, avšak problém je překlopen do oblasti eliptických křivek.

Na závěr bylo provedeno praktické měření doby autentizace tímto protokolem s různým typem eliptických křivek. Podle předpokladu vychází nejrychleji křivka secp192k1 s nejnižší bitovou délkou (192 bitů). Průměrný čas autentizace byl 210,9 ms (pouze telefonem) a 805,5 ms (telefon i hodinky).

Nejpomaleji probíhala autentizace v případě křivky prime256v1. I přes srovnatelnou bitovou délku (256 bitů) s křivkou secp256k1 byla v extrémních případech doba autentizace až dvakrát delší. Průměrný čas autentizace byl 542,2 ms (pouze telefonem) a 1891,4 ms (telefon i hodinky). Dle simulace je ve všech případech patrná proměnlivá odchylka 500–800 ms, způsobená především vzájemnou komunikací zařízení a čekáním na provedení operace jiným zařízením.

I přes určité výchylky při měření je dosažená průměrná doba autentizace pro uživatele stále přijatelná vzhledem k bezpečnosti, kterou kryptosystém poskytuje. Vytvořený systém aplikací by mohl být například využitý v přístupových a jiných systémech vyžadující ověření identity.

LITERATURA

[1] BURDA, Karel. *BEZPEČNOST INFORMAČNÍCH SYSTÉMŮ*. Brno: FEKT VUT Brno, 2005, 104 s.

[2] STANFORD-CLARK, Andy. *MQTT V3 Protocol Specification* [online]. IBM, 1999 [cit. 2016-12-12]. Dostupné z: <http://stanford-clark.com/MQTT/>.

[3] Využití protokolu MQTT nejen pro M2M a IoT. In: *Blog Břicháček* [online]. 2015 [cit. 2016-12-05]. Dostupné z: <https://blog.brichacek.net/vyuziti-protokolu-mqtt-nejen-pro-m2m-a-iot/>.

[4] Protokol MQTT: komunikační standard pro IoT. In: *ROOT.cz* [online]. 2016 [cit. 2016-12-05]. Dostupné z: <https://www.root.cz/clanky/protokol-mqtt-komunikacni-standard-pro-iot/>.

[5] PAWANI, Porambage, Schmitt CORINNA, Kumar PARDEEP, Gurtov ANDREI a Ylianttila MIKA. PAuthKey: A Pervasive Authentication Protocol and Key Establishment Scheme for Wireless Sensor Networks in Distributed IoT Applications. *International journal of distributed sensor networks*. 2014. DOI: 10.1155/2014/357430. ISSN 1550-1329. Dostupné také z: <https://www.hindawi.com/journals/ijdsn/2014/357430/abs/>.

[6] SHELBY, Zach, Klaus HARTKE a Carsten BORMANN. *Constrained Application Protocol (CoAP)* [online]. 2013, , 1-118 [cit. 2016-12-05]. Dostupné z: <https://tools.ietf.org/html/draft-ietf-core-coap-14>.

[7] KOTHMAYR, Thomas, Corinna SCHMITT, Wen HU, Michael BRÜNIG a Georg CARLE. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. *37th Annual IEEE Conference on Local Computer Networks - Workshops*. IEEE, 2012,str. 956-963. DOI: 10.1109/LCNW.2012.6424088. ISSN 978-1-4673-2129-7.

[8] RESCORLA, Eric a Nagendra MODADUGU. *Datagram Transport Layer Security Version 1.2* [online]. RFC 6347, 2012 [cit. 2016-12-08]. Dostupné z: <https://rfc-editor.org/rfc/rfc6347.txt>.

[9] HAJNÝ, Jan, Petr DZURENDA a Lukáš MALINA. Multi-Device Authentication using Wearables and IoT. *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications - Volume 4: SECRYPT*. 2016, str. 483-488. DOI: 10.5220/0006000004830488. ISSN 978-989-758-196-0.

[10] What It Does. In: *NFC Forum* [online]. NFC Forum, © 2013 [cit. 2016-12-05]. Dostupné z: <http://nfc-forum.org/what-is-nfc/what-it-does/>.

[11] DZURENDA, Petr, Jan HAJNÝ a Lukáš MALINA. Chytré telefony jako náhrada čipových karet. In: *SystemOnLine* [online]. 2014 [cit. 2016-12-05]. Dostupné z: <https://www.systemonline.cz/it-security/chytre-telefony-jako-nahrada-cipovych-karet.htm>.

[12] Stačí přiložit: NFC a jeho využití v praxi. In: TRČÁLEK, Antonín. *Mobilmania.cz* [online]. 2013 [cit. 2016-12-05]. Dostupné z: <http://www.mobilmania.cz/clanky/staci-prilozit-nfc-a-jeho-vyuziti-v-praxi/sc-3-a-1325034/default.aspx>.

[13] Near Field Communication. In: *Android developer portal* [online]. © 2014 [cit. 2016-12-05]. Dostupné z: <https://developer.android.com/guide/topics/connectivity/nfc/index.html>.

[14] Tag Type Technical Specifications. In: *NFC Forum* [online]. NFC Forum, © 2013 [cit. 2016-12-05]. Dostupné z: <http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/tag-type-technical-specifications/>.

[15] Host-based Card Emulation. In: *Android developer portal* [online]. © 2014 [cit. 2016-12-05]. Dostupné z: <https://developer.android.com/guide/topics/connectivity/nfc/hce.html#SecureElement>.

[16] JEPSON, Brian, Don COLEMAN a Tom IGOE. *Beginning NFC*. O'Reilly Media, 2014, 246 s. ISBN 978-1-4493-7206-4.

[17] *Application Protocol Data Unit: Smart Card Application Development* [online]. SIEVÄNEN, Markk. 2003 [cit. 2016-12-05]. Dostupné z: <http://www.tml.tkk.fi/Studies/T-110.497/2003/lecture4.pdf>.

[18] *INTERNATIONAL STANDARD ISO/IEC 7816-4: Identification cards — Integrated circuit cards — Part 4: Organization, security and commands for interchange*. 2. vydání. © 2005.

[19] GOMEZ, Carles, Josep Paradells a Joaquim OLLER. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors*. 2012, **12**(9). DOI: 10.3390/s120911734. ISSN 1424-8220.

[20] Bluetooth Low Energy (Bluetooth LE). In: ROUSE, Margaret. *IoT Agenda* [online]. 2014 [cit. 2016-12-05]. Dostupné z: <http://internetofthingsagenda.techtarget.com/definition/Bluetooth-Low-Energy-Bluetooth-LE>.

[21] Bluetooth Low Energy. *Android developer portal* [online]. © 2014 [cit. 2016-12-05]. Dostupné z: <https://developer.android.com/guide/topics/connectivity/bluetooth-le.html>.

[22] Bluetooth Vs. Bluetooth Low Energy: What's The Difference? In: *LinkLabs* [online]. © 2015 [cit. 2016-12-05]. Dostupné z: <https://www.link-labs.com/bluetooth-vs-bluetooth-low-energy>./

[23] TOWNSEND, Kevin., Robert. DAVIDSON, AKIBA. a Carles. CUFÍ. *Getting started with Bluetooth low energy: tools and techniques for low-power networking*. Revised First Edition. Sebastopol, CA: O'Reilly, 2014. ISBN 9781491949511.

[24] BARKER, Elaine. *Digital Signature Standard*. Gaithersburg: National Institute of Standards and Technology, 2013. ISBN 120921480-2480-01. Dostupné z: <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>.

[25] BROWN, Daniel R. L. *Standards for Efficient Cryptography: SEC 2: Recommended Elliptic Curve Domain Parameters*. 2. Certicom, 2010. Dostupné z: <http://www.secg.org/sec2-v2.pdf>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

IoT	Internet věcí – Internet of Things
MITM	Man-in-the-middle – útok na kryptografii
PKI	Public Key Infrastructure
NFC	Near Field Communication
BLE	Bluetooth Low Energy
HDM16	Multidevice protokol – Hajný, Dzurenda, Malina, 2016
PIN	Personal Identification Number
SMS	Short Message Service
MQTT	Message Queuing Telemetry Transport
IBM	International Business Machines Corporation
TCP/IP	Transmission Control Protocol/Internet Protocol
M2M	Machine to machine
MQTT-SN	MQTT for Sensor Networks
SSH/TLS	Secure Sockets Layer/Transport Layer Security
LAN	Local Area Network
AWS IoT	Amazon Web Services IoT
PAuthKey	Pervasive Authentication Protocol and Key Establishment
WSN	Wireless Sensor Networks – Bezdrátová síť senzorů
CA	Certifikační Autorita
DTLS	Datagram Transport Layer Security
CoAP	The Constrained Application Protocol
ROM	Read Only Memory
RAM	Random Access Memory
URI	Uniform Resource Identifier

HTTP	Hypertext Transfer Protocol
UDP	User Datagram Protocol
HMAC	Hash-based Message Authentication Code
CBC	Cipher-Block Chaining
ECC	Elliptic curve cryptography
RSA	Šifra s veřejným klíčem – iniciály autorů Rivest, Shamir, Adleman
PKC	Public-Key Cryptography – Kryptografie s věřejným klíčem
DoS	Denial of Service
RFID	Radio Frequency Identification
NDEF	NFC Data Exchange Format
ISO/IEC	International Organization for Standardization (ISO) and the International Electrotechnical Commission
NDEF	NFC Data Exchange Format
FeliCa	Felicity Card
HCE	Host-based Card Emulation
APDU	Application Protocol Data Unit
TNF	Type Name Format
MIME	Multipurpose Internet Mail Extensions
NFC RTD	NFC Record Type Definition
TPDU	Transmission Protocol Data Unit
AID	Application ID
PAN	Personal Area Network
MLDP	Microchip Low-energy Data Profile
CRC	Cyclic Redundancy Check
EDR	Enhanced Data Rate

GAP	Generic Access Profile
GATT	Generic Attribute Profile
ATT	Attribute Protokol
SIG	Special Interest Group
UUID	Universally Unique IDentifier
SIG	Special Interest Group
NIST	National Institute of Standards and Technology
SECG	Standards for Efficient Cryptography Group

SEZNAM PŘÍLOH

A PŘÍLOHA - ZDROJOVÉ KÓDY

Do systému jsou vloženy pouze zdrojové kódy aplikací. Z důvodu úspory místa nejsou vloženy celé projekty ani externí knihovny. Všechny potřebné části jsou uložené na připojeném disku DVD.

- `/` ..... Systémový adresář
  - `HDM16_Verifier` ..... Zdrojové kódy aplikace pro PC
  - `HDM16_Mobile` ..... Zdrojové kódy aplikace pro telefon
  - `HDM16_Wear` ..... Zdrojové kódy aplikace pro hodinky

B OBSAH PŘILOŽENÉHO DVD

Na DVD jsou uloženy zdrojové kódy a externí knihovny výstupní aplikací.

- `/` ..... kořenový adresář přiloženého DVD
  - `workspace_eclipse` ..... Projekty vytvářené v Eclipse
    - `HDM16_Verifier` ..... Aplikace pro PC
  - `workspace_Android_Studio` ..... Projekty vytvářené v Android Studiu
    - `HDM16_Mobile` ..... Aplikace pro telefon
    - `HDM16_Wear` ..... Aplikace pro hodinky
  - `knihovny` ..... Externí knihovny
    - `prov-1.54.0.0` ..... Knihovna pro práci s EC – 01
    - `core-1.54.0.0` ..... Knihovna pro práci s EC – 02
    - `smartcardio etc.` ..... Knihovna pro čtečku karet